



Departament de Llenguatges i Sistemes Informàtics  
UNIVERSITAT POLITÈCNICA DE CATALUNYA

## **Master in Computing**

### **Master of Science Thesis**

# **ANALYSIS AND COMPARISON OF FACIAL ANIMATION ALGORITHMS: **CARICATURES****

Anna Isabel Bellido Rivas

Advisor/s: Pere-Pau Vázquez Alcover

8 de setembre de 2010

## Index of contents

1. Introduction (3)
2. Classification of different methods to generate caricatures (4)
  - a. First group of algorithms: Tools for caricaturists (5)
  - b. Second group of algorithms: Caricaturists talent (8)
  - c. Third group of algorithms: Data Bases approaches (9)
3. First method: A Method for Deforming-driven Exaggerated Facial Animation Generation (21)
  - a. RBF (21)
  - b. Explanation of the method (23)
  - c. Implementation and results (25)
  - d. Advantages of the method (28)
  - e. Drawbacks and complaints (29)
4. Second method: Exaggeration of Facial Features in Caricaturing (32)
  - a. Bézier curves (32)
  - b. Explanation of the method (33)
  - c. Implementation and results (35)
  - d. Advantages of the method (42)
  - e. Drawbacks and complaints (42)
5. Third method: Facial Caricature Generation Using a Quadratic Deformation Model (43)
  - a. Rubber-sheet transformations (43)
  - b. FACS (45)
  - c. Pre-paper: Facial Expression Representation Using a Quadratic Deformation Model (47)
  - d. Explanation of the method (49)
  - e. Implementation and results (50)
  - f. Advantages of the method (60)
  - g. Drawbacks and complaints (60)
6. Conclusions (63)
7. Bibliography

# 1. Introduction

Caricatures have always been a funny way of creating drawings of people emphasizing the prominent features of the face.

*Definition of “caricature” and “exaggerate” according to the Merriam Webster dictionary:*

car·i·ca·ture

*noun*

\ˈker-i-kə-ˌchür, -ˌcher, -ˌtyür, -ˌtür, -ˈka-ri-\

- a. A representation, especially pictorial or literary, in which the subject's distinctive features or peculiarities are deliberately exaggerated to produce a comic or grotesque effect.*
- b. The art of creating such representations.*

ex·ag·ger·ate

*verb* \ig-ˈza-jə-,rāt\

- a. To enlarge beyond bounds or the truth.*
- b. To enlarge or increase especially beyond the normal*

Artists throughout the world have the ability to detect those features among the others. This is a skillful task as not everyone has the expertise to observe a face and seeing those little nuances that stand out. Apart from that, seeing those characteristics does not guarantee that a caricature is well done, but drawing skills are also necessary to create a successful caricature.

There are three essential elements in caricaturing that must be taken into account:

1. **Exaggeration:** An artist decides what features to exaggerate and the scaling factor
2. **Likeness:** Visual similarity between the original face and the exaggerated face.
3. **Statement:** Addition of some personality by editorializing the caricature.

Nevertheless, people's tolerance to caricatures is very loose, then, it seems like any creation of a drawing that resembles the original person and looks funny can be easily accepted as a true caricature.

This seems to have led to change the orientation of production of a large number of papers that have been published lately. Many papers are now focused on creating drawings/modified photographs of people changing the original shape. Some, however, still keep the previous idea of emphasizing in a drawing people's facial characteristics, making line drawings sometimes decorated with a bit of color or some background.

The thesis will be aimed to review what has been done from around the 2000 until now regarding the caricature generation field in 2D.

It will be organized in classifying the methods found first, telling their contributions to the field and choosing a paper among them to implement and discuss more thoroughly. A total of three papers will be selected.

Finally, an overview discussion on the papers implemented and their contributions to the field will be given.

*Brief comment on the Master Thesis small change of title:*

In the very beginning, when I was planning to do the thesis, I talked with my tutor and found that doing a review and comparison of some methods in the facial animation field would suit. However, while reading papers on the topic, I found that a great number of them required hardware which I didn't have any access to. The generation of 2D caricatures is still close to the field, and it didn't need any additional hardware devices.

## 2. Classification of different methods to generate caricatures

As I mentioned previously, it's not easy to generate a caricature from an image. First of all, you must be able to detect what to exaggerate, and then, create the picture. Once the facial features to enhance have been chosen, the process may be straight forward in a sense; that is, right when you know exactly what to do in terms of goal and tools to achieve it, computer scientists unavoidably start thinking that they can fully automate the process. This leads to a bunch of tools that help the caricaturists to finish their artistic creations. I will call this the first group of caricatures generation.

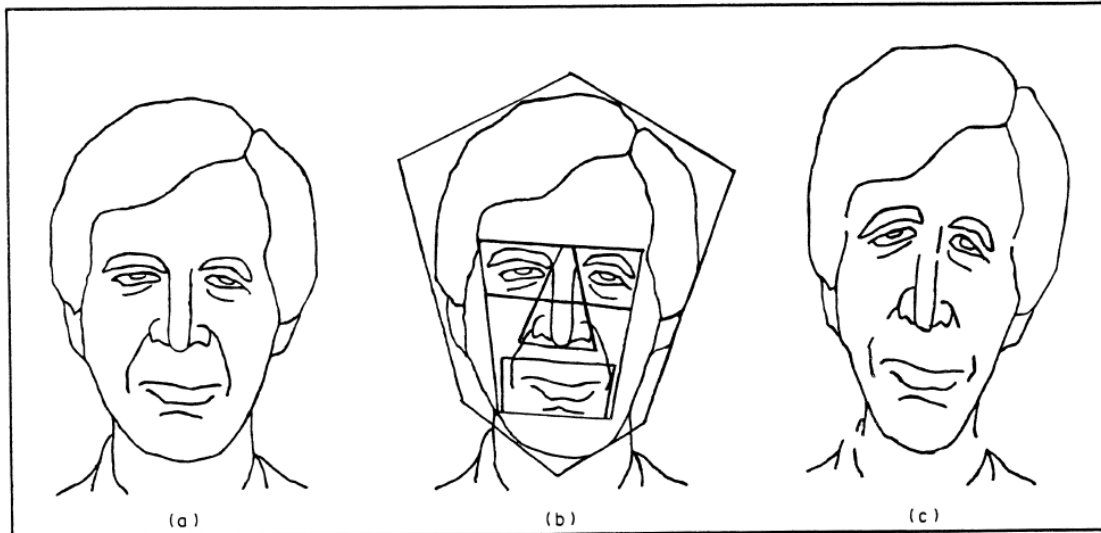
Some artists are not happy at all with these toolboxes as this undermines their artistic skills, as the set of possible deformations end up being quite limited. Thus, the second group of caricatures generation tools appear. This group is formed by those that leave the responsibility of choosing how to trace every line in a drawing to the artists. However, this group is almost completely manual by nature but computerized at the same time.

According to the definition of exaggeration, emphasizing a feature means that you know exactly what should be normal and that you see that this concrete facial feature is beyond those normal terms. But, what is normal? What can be considered as a standard face? Some papers in the former groups have introduced the idea of an input mean face, but this is only used as a starting point.

Trying to answer this question appears the third group of algorithms. This group is mainly formed by those that keep large data bases of people's faces which are described differently depending on the algorithm, although they all have in common that they intend to define what a standard face is. Once this is determined, comparing every input face with the "normal" one makes them choose what to exaggerate. However, there's still a problem to be addressed: how do you exaggerate a feature? In this group there are also some other algorithms that make use of data bases of artists that, from a normal face which is stored too, they see which are the facial characteristics of a face and they deform them according to their understanding/methodologies. Those changes in the drawings are stored to lead future facial deformations.

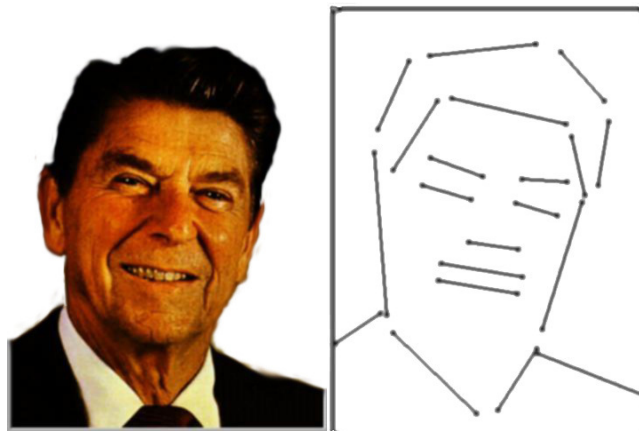
## 2A. First group of algorithms: Tools for caricaturists

This group is where the first algorithm on caricatures generation belongs to. It was “*Caricature Generation: The Dynamic Exaggeration of Faces by Computer*” by Susan E. Brennan in 1982 [1]. It consisted on a method to perform an initial sketch of a caricature.



This one, which is never accurate, must be modified by the user observing the outstanding characteristics of the face of the person, and modifying them following his/her own criteria.

Another method appeared quite later by Akleman in 1997 called “*Making caricature with morphing*” [2]. The author interpreted that not everyone can capture the essence of a face as a gifted caricaturist can, but they do can modify a very simple scheme that has been already preconstructed, people find it easier to exaggerate a face. A scheme like this one is given:



And the user must only modify the sketch. A warping method to exaggerate the face is performed after that, regarding the changes done in the sketch.

This last method was followed by one called “*Facial caricaturing with motion caricaturing in PICASSO system*” by Tominaga et. al. in 1999 [3]. This one was a template-based approach. They computed an average face first and according to the differences they found regarding their templates, an initial moderate sketch is created. The user can later on modify the drawing.

Bearing in mind that in the early 2000 a lot of methods with large data bases came up, only other methods modifying the warping process were published. This is the case of the first implemented algorithm called “*A Method for Deforming-driven Exaggerated Facial Animation Generation*”, by Wei et. al. in 2008 [5]. In this case, the warping tool used are RBFs (Radial Basis Functions). Further information will be given in the chapter describing the implementation, results and conclusions on this paper.

## *2B. Second group of algorithms: Caricaturists talent*

This group is formed by those that offer the minimum amount of suggestions to the caricaturists. This is almost like letting the caricaturist do the work himself/herself. For this reason, not many papers have been published in this direction, as if caricaturists are not going to be helped considerably, they will reasonably prefer going on working with a pencil and a sheet of paper.

However, there are still some algorithms that produce nice drawings like the following two offering only a little almost negligible help:

In 2004, a paper called “*Human facial illustrations: Creation and psychophysical evaluation*” by Gooch was published [4]. It offered a method to create a black and white initial image from a photograph of a face. Moreover, the system required a lot of interaction with a user to produce the final caricature. This approach required severe professional skills.

The other method is the second implemented approach, which was published in 2006 by Luo et. al. and its title was “*Exaggeration of Facial Features in Caricaturing*” [6]. Further details will be given in its chapter.



## 2C. Third group of algorithms: Data Bases approaches

This group is the biggest one due to its appeal. What data bases supply are means of virtually getting closer to generality. They provide a statistical mattress with which the authors of the papers form models of what could be considered as standard values. Fake or not, these models are supported by the tones of subjects included, thus, valid for a large number of situations. However, these models are always limited to the set of data stored, and this may not be always sufficient, because a new datum may be given which cannot be expressed with the model created.

As it has been said, these models are basically created for the two main phases of the generation of a caricature: the prominent facial features detection and the exaggeration of these.

Regarding that the papers in this group are the most recent ones, I will explain most of them more in depth.

The first one of the early 2000 that I could find was “*Example-based facial sketch generation with non-parametric sampling*” by Liang in 2001 [7].

The authors of this paper hired an artist who drew lots of sketches of caricatures following a concrete style. Given an input image transformed into a sketch, they try to find the sketches stored in the data base that most closely resemble it and apply the transformation done to the neighbors to the given one.

Right after this one, in 2002, another paper similar to the former was published by the same author. Its title is “*Example-based Caricature Generation with Exaggeration*” by Liang [8].

This method uses a data base of 184 images; a half is the original faces and the other half is the corresponding caricature, all of them organized in pairs. All the caricatures were drawn by the same artist in order to keep the same style among the drawings. The feature points of every face were labeled manually.

The authors decouple all the faces in the model in two parts:

- Shape : the set of feature points of a face aligned to a standard (MeanShape)
- Texture: a shape-free image, warped to a standard one (MeanShape)

This way, the deformation process of a face is also decoupled in shape exaggeration and texture style transferring.

Now, they have the data base already built.

Given an input image, they extract the feature points and use the Shape Exaggeration Model they created.

To build it they considered three options:

- a. Scaling the difference:

Considering  $S$  a given shape, we know that:

$$S = S_{mean} + \Delta S$$

Being  $\Delta S$  the difference between the given shape and the mean. Then:

$$\Delta S' = b * \Delta S$$

This means that the final face is the result of scaling every difference from the mean shape. However, this produces errors, as the mean shape is unique, and some features that may not need to be exaggerated, end up enhanced with this.

b. K – Nearest Neighbors (KNN):

The main goal in here is to express the difference  $\Delta S$  with some of the shapes in the data base, concretely, those that are closest to the input shape.

$$\Delta S_0 = [\Delta S_{01} ; \Delta S_{02} \dots \Delta S_{0k}] w$$

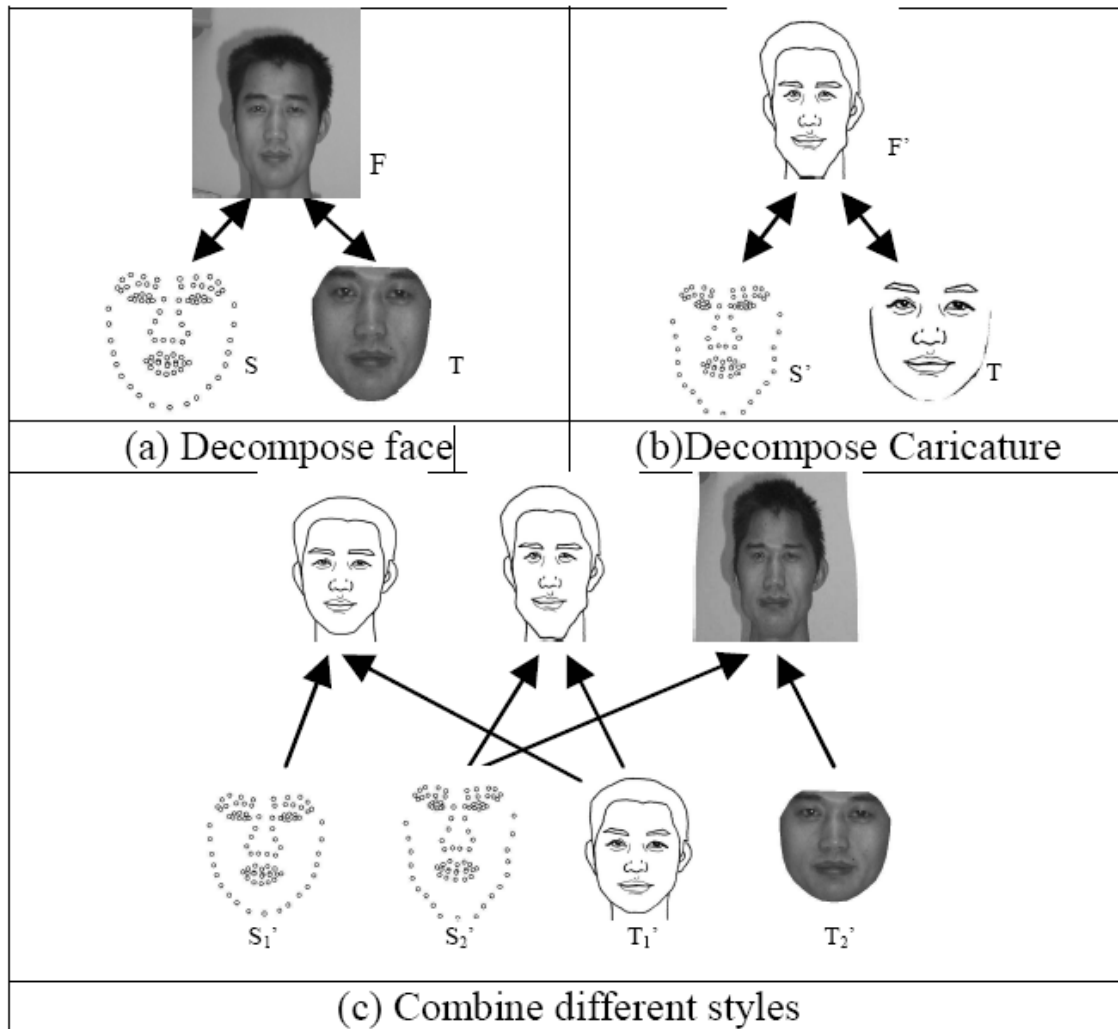
This interpolation is done with weights found through least squares. Even though this one looks much more reliable, there's a considerable neglectance of the facial subtleties.

c. Example-based approach:

In this case, the data base is organized in prototypes of exaggeration directions; this way, those subsets can be used together avoiding the smoothing process of averaging exaggeration features that have nothing to do with each other.

The example-based approach, as the very title of the paper suggests, was the chosen one. Once a subset of prototypes has been chosen to exaggerate a face, the shape exaggeration is done and a warping process of the original texture of the input face is done Thin Plate Splines.

A graphic example of the whole algorithm goes as follows:



Surprisingly, in 2004 another paper was published called “*Automatic Caricature Generation by Analyzing Facial Features*” by Chiang [9]. This one works out the average shape and exaggerates the difference, producing worse results than the previous paper from the 2002.

In 2006, Chen et. al. published the paper “*Generation of 3D Caricature by fusing Caricature Images*” [10].

This paper is similar to the previous but adding 3D. It uses the principles of stereoscopy to create a 3D caricature from two 2D caricatures.

This paper make use of MPEG-4: in order to find a total of 119 nodes in the face (facial features) they use AAM (Active Appearance Models) [11], but they make the most of FDPs and FAPs to control the shape and appearance of the facial features. Those features are then classified into eight different groups and organized at the same time in 3 hierarchies.

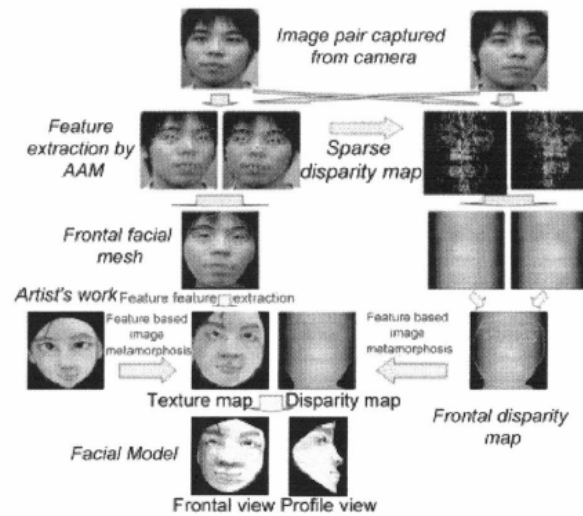
They store a data pool of faces used to compute a mean shape for every facial component, a total of 53 people from 2 views (frontal and side views).

Then, every input face is again extracted the mean shape, and that difference is the candidate to be exaggerated for every facial component.  
The caricature generation is performed through warping the final image regarding the transformed feature points.

This is done for the 2 input faces (from frontal and side views), they build the 3D face. It is done in 3 phases:

1. Preprocessing: region segmentation from a set of control points
2. Feature correspondence: 5x5 windows of tolerance for every point to find the best match in the other image
3. Depth map recovery: principle of binocular fusion.

An overview of the system can be seen below:



In the same year “*Mapping Learning in Eigenspace for Harmonious Caricature Generation*” came up, by Liu [12].

In this algorithm, some treatment on the store data is done: dimension reduction, concretely, a PCA (Principal Components Analysis) [13].

It’s organized in two parts: a mapping learning in eigenspace and caricature generation.

The mapping learning part focuses on building a good data base with the PCA:

They grab two sets  $X_a$  and  $X_b$ . The first represents the original shape and the second the target one.

First the average:

$$\bar{X} = \frac{1}{N} \sum_{i=1}^N X_i$$

Then the differences:

$$\Phi_i = X_i - \bar{X}$$

After that, the covariance matrix:

$$C = \frac{1}{M} \sum_{i=1}^M \Phi_i \Phi_i^T$$

With this symmetrical matrix C (thus, always invertible), we compute the diagonal form of it. The eigenvectors associated to the biggest eigenvalues express most of the shape. In order to know how many values to store while not losing a severe amount of information, we give a ratio R, the k eigenvectors must fulfill:

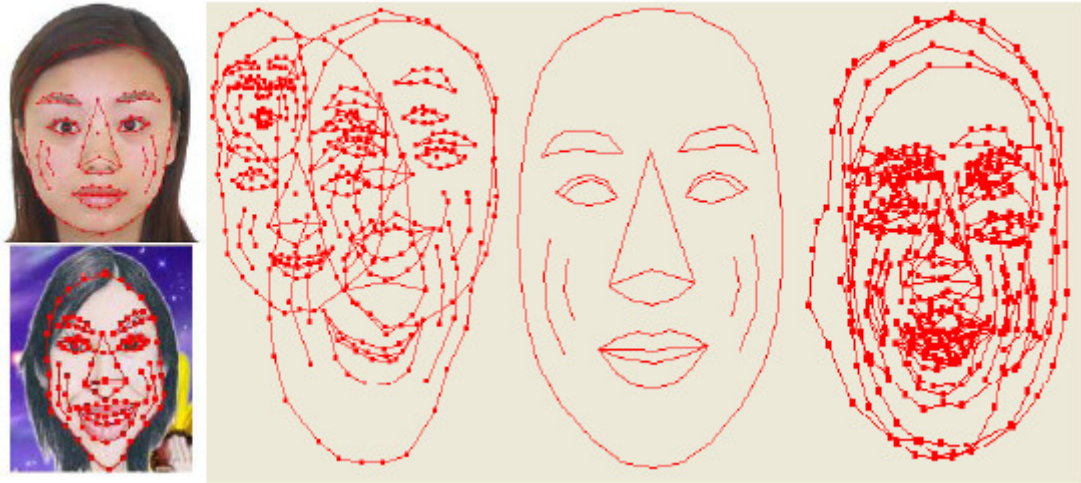
$$\sum_{i=0}^k \lambda_i / \sum_{i=0}^{M-1} \lambda_i \geq R$$

Now, from those eigenvectors we can obtain the transformation of the initial set  $X_a$  into the space of the diagonalization to obtain  $w_a$  and  $w_b$ :

$$\omega_a = \mu^T (X_a - \bar{X})$$

$$\omega_b = \mu^T (X_b - \bar{X})$$

This is the data process of this paper:



Once this is built, the authors try to find a relation between these two groups in order to use it when applying the deformations to new input. The idea is to construct a regression function that correlates both sets:

$$f(\omega) = \sum_{i=1}^l \beta_i K(\omega_i, \omega) + b$$

where  $K(\omega_i, \omega_j) = \exp\left(-\frac{\|\omega_i - \omega_j\|^2}{2\sigma^2}\right)$

is a kernel function satisfying Mercer's conditions

$\beta_i$  is the coefficient of the i-th sample

$w_i$  is the input

$f(w_i)$  is the output

This can be used to predict the deformation regarding the input image.  
Given an input  $X$ , we first must find its position in the eigenspace of the faces in the data base:

$$\omega = \mu^T (X - \bar{X})$$

In this space, we know where the input difference must go, as it's determined by function  $f$ :

$$\omega' = f(\omega)$$

Finally, adding the mean shape to the new difference back in the original space with  $\mu$ , we get the deformation:

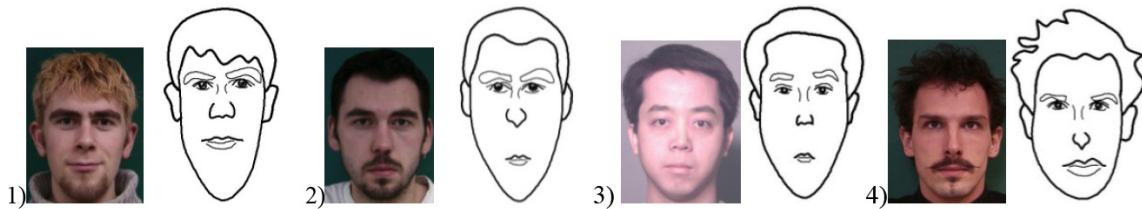
$$X' = \bar{X} + \mu\omega'$$

A paper that also transforms the data base into eigenspace goes after this one. In 2009, Chen et. al. published another paper using PCA called “*Example Based Caricature Synthesis*” [14].

The analysis of the data base is equivalent to the one in the previous paper. However, in this case, they decide to choose different artists with different styles and deal with harmony between them exploiting the “likeness” characteristic of a good caricature showing a measure to capture it. This measure is the Modified Hausdorff Distance (MHD) [16]. This way, they can obtain the visual similarity (likeness) and modify slightly the final caricature to resemble more the original image.

They mention the fact that the facial features cannot be analyzed and transformed fully separately, as they are interrelated fundamentally. For example: T-rule used to draw caricatures [15].

Generated results:



In this very year, “*Application of Facial Feature Localization using Constrained Local Models in Template-based Caricature Synthesis*” by Lei et. al. [17].

In this approach, the authors do the analysis of the facial component according to the Chinese physiognomy and they use a decision tree classification algorithm to obtain the mapping between facial attributes and categories. They construct some patterns to apply to form a caricature, and it is done for every facial feature.

A preprocess of faces alignment is needed in this method. They do it using a Constrained Local Model which consists of two parts:

- Model shape variation
- Variation in appearance of a set of local support regions

We will focus on the shape variation model. We start with an initial shape with  $n$  landmarks  $\{ (x_i, y_i) \}$ . These landmarks can be represented in a  $2n$  vector  $\mathbf{x}$ :

$$\mathbf{x} = (\mathbf{x}_1^T, \dots, \mathbf{x}_n^T)^T, \mathbf{x}_i = (x_i, y_i)^T$$

The non-rigid shape transformation can be expressed as a point distribution model:

$$\mathbf{w}(\mathbf{x}; \mathbf{p}) = \mathbf{x} + \mathbf{V}\mathbf{p}$$

where  $\mathbf{p}$  is a parametric vector describing the nonrigid warp  
 $\mathbf{V}$  is a matrix of eigenvectors with a 5% data loss

In order to force a global 2D similarity transformation, they force the first 4 eigenvectors to correspond to similarity variation [18]. For instance:

If we have a base mesh:

$$\mathbf{s}_0 = (x_1^0, y_1^0, \dots, x_n^0, y_n^0)$$

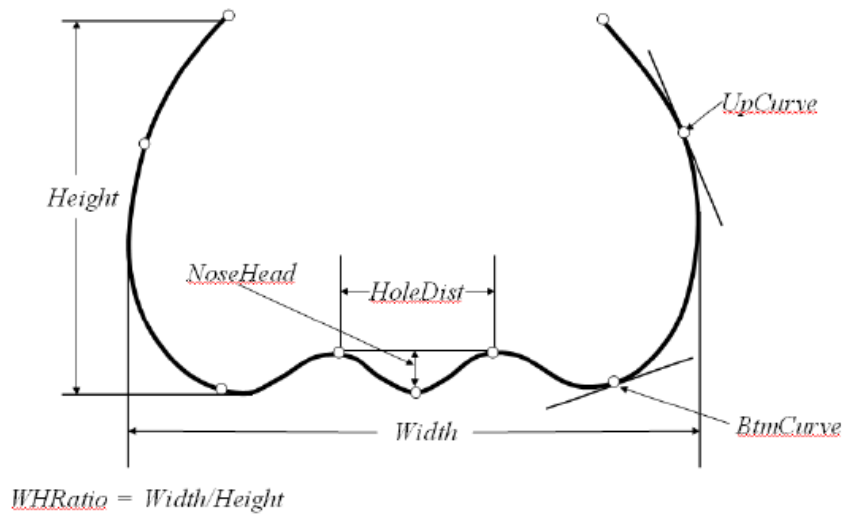
Then

$$\begin{aligned} \mathbf{s}_1^* &= (x_1^0, y_1^0, \dots, x_n^0, y_n^0) \\ \mathbf{s}_2^* &= (-y_1^0, x_1^0, \dots, -y_n^0, x_n^0) \\ \mathbf{s}_3^* &= (1, 0, \dots, 1, 0) \\ \mathbf{s}_4^* &= (0, 1, \dots, 0, 1) \end{aligned}$$

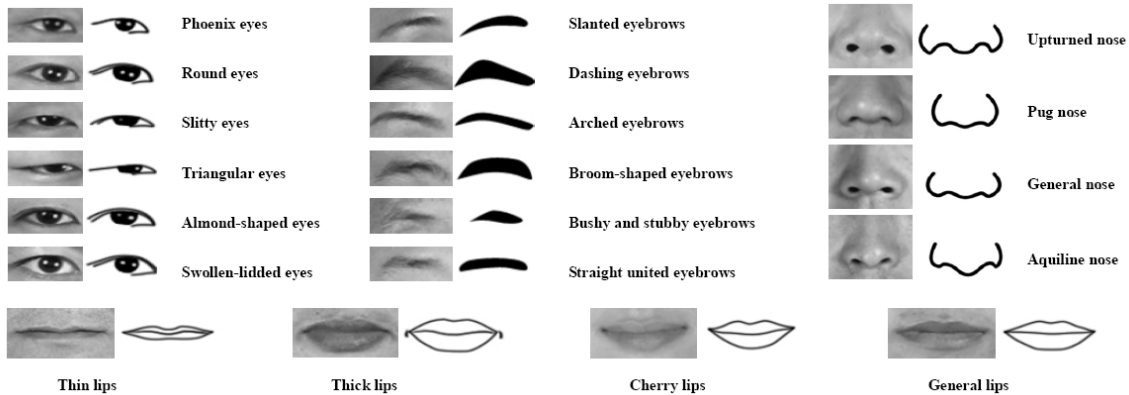
So as to obtain a reasonable shape, they constrain  $\mathbf{p}$  to live in a hyperellipsoid:

$$\left( \sum_{i=1}^t \frac{p_i^2}{\lambda_i} \right) \leq M_t$$

In the process of creation of the caricature, given an image of a face, they first select the discrimination features to be caricaturized, bearing in mind characteristics like slope, curvature, aspect ratio, size, specific shape... checking the image like in this image:



Now, for every feature, they must find the appropriate pattern to apply among those created previously:



Finally, they have a set of patterns to apply and different geometrical locations, then, the only think to do is to assemble everything in a line drawing.

The last remarkable paper I found in this year has the title “*Semi-supervised Learning of Caricature Pattern from Manifold Regularization*” by Liu et. al [21]. This method is focused on the prediction of the pattern to apply to an image to create the caricature.

The authors base their motivation on the results of the paper “*Nonlinear dimensionality reduction by locally linear embedding*” [22] and those in the paper “*Mapping Learning in Eigenspace for Harmonious Caricature Generation*” explained earlier. In this last paper, it’s shown that it’s feasible to construct a PCA model with up to 17 eigenvectors describing the morphing between original shapes and caricatures. Apart from that, in the first paper the authors show that a manifold, a nonlinear model, is suitable to express the face distribution.



Briefly, they treat the data base of images and their corresponding caricatures to low dimensional caricature patterns. Considering that it is very hard to translate a caricature pattern into numbers, they prefer looking for the distribution of the patterns, building a continuous pattern map. The process is described below:

First of all, they detect the shape of the faces from photographs using Active Shape Models (ASM) [23]. This is a training algorithm through a data base, then, it's only suitable to detect the faces of the original photographs while useless for the caricatures, what is not mentioned by the authors who later presume that they also have the shape of those. The total of caricatures is 1027 and the total of faces is 1841.

Right after that, they align all the shapes to a mean shape that they previously calculate, considering it to be the standard scale. In order to do this, in a similar way as the authors of the paper "*Application of Facial Feature Localization using Constrained Local Models in Template-based Caricature Synthesis*" explained before did, they use two matrices to apply the affine transformation of alignment:

$$A_i = \begin{bmatrix} x_{i0} & -y_{i0} & 1 & 0 \\ y_{i0} & x_{i0} & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_{i(n-1)} & -y_{i(n-1)} & 1 & 0 \\ y_{i(n-1)} & x_{i(n-1)} & 0 & 1 \end{bmatrix}_{2n \times 4} \quad Z = \begin{bmatrix} s \cos \theta \\ s \sin \theta \\ t_x \\ t_y \end{bmatrix}$$

being  $X_i$  the group of shapes formed with  $\{x_{i0}, y_{i0}, \dots, x_{i(n-1)}, y_{i(n-1)}\}$   
 $n$  is the number of shapes available  
 $s$  is the scaling coefficient  
and  $\theta$  is the rotation angle

The way to find  $Z$  is through  $A_i$  and the mean shape:

$$\bar{X} = A_i Z \Rightarrow Z = (A_i' A_i)^{-1} A_i' \bar{X}$$

Once  $Z$  is found, they can align the images:

$$X_i' = A_i Z = A_i (A_i' A_i)^{-1} A_i' \bar{X}$$

Now, all the shapes are aligned to the mean shape.

Subsequently, the authors apply a dimension reduction to the acquired shapes using LLE. According to LLE, every face (even caricatures) can be expressed easily as the linear combination of the  $K$ -nearest neighbors. The faces can be reconstructed through weight matrices, in a similar way as the interpolation methods do.

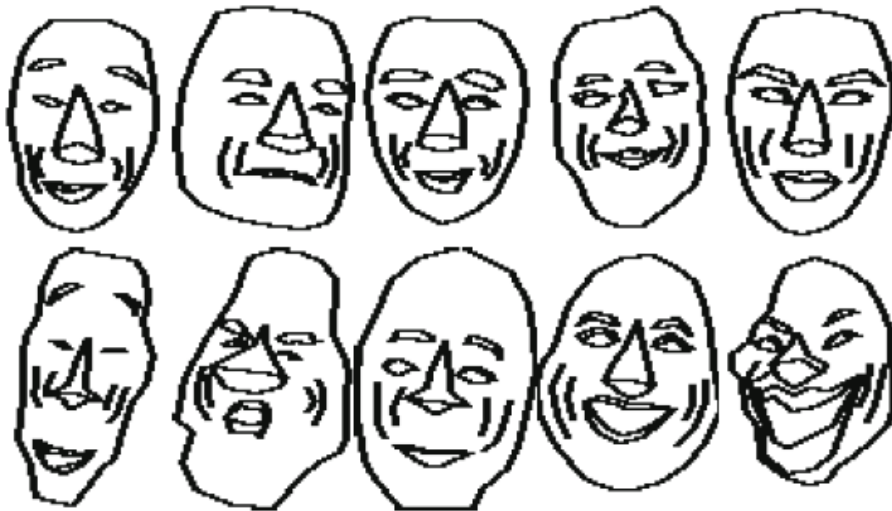
However, we must always take into account that there's an error measure:

$$\varepsilon(W) = \sum_i |X_i - \sum_j W_{ij} X_j|^2$$

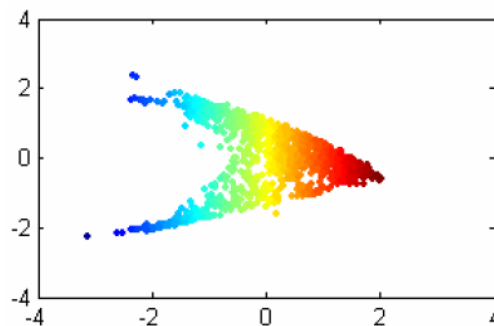
The constraints they use to build the weights matrix are the fact that the sum of a row must be 1, and the weight must be 0 when  $X_j$  is not one of the chosen neighbors for  $X_i$ .

Then, they build the manifold regularization, what consists on finding a way to describe a sparse set of data. The same would do any interpolation/approximation method, to be more clear. The regularization they use is detailed in the paper “*Manifold Regularization: a Geometric Framework for Learning from Labeled and Unlabeled Examples*” [24].

However, they can’t apply it straight forward to the large amount of data that they have (1841 original images and 1027 caricatures), not even making reliable matchings between them. They end up with a total of 198 well-prepared pairs like these:

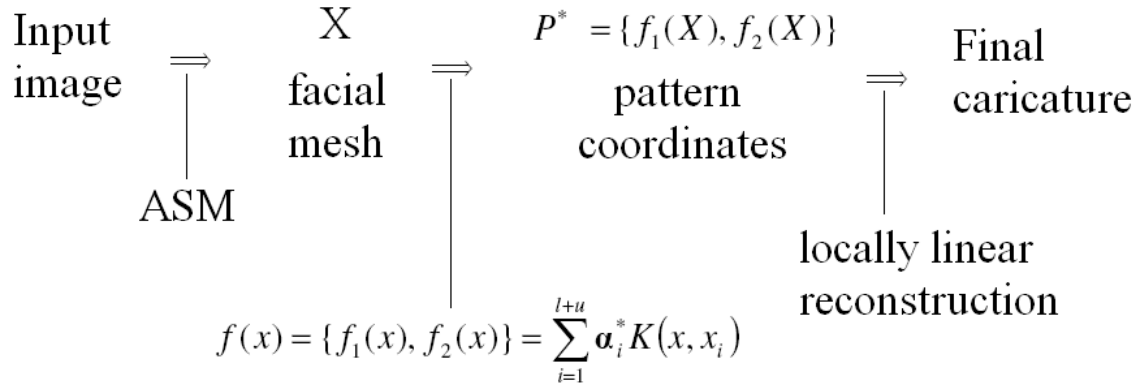


Immediately can a manifold learning function be applied to this data based on MR (semi-supervised learning method). The final data is structured as follows:



Briefly, what they have up to now is a continuous manifold of transformation patterns obtained after aligning them and reducing their dimensionality. Therefore, what comes next is talking about how to apply this manifold to do the prediction of the exaggerated shapes.

The process can be seen in this scheme:



The first, second and third steps have been previously addressed, and the forth step will be explained below:

We now have the pattern coordinates  $P^*$ , but we must reverse the process of dimension reduction and manifold structure creation to obtain a final caricature shape.

The process starts with  $Y'$  shape (dimension reduced and treated). We must find the  $k$  neighbors  $Y'_i$ . Then, they compute the weights matrix in order to build the reconstruction through those weighted neighbors, through minimizing the error equation given.

Every  $Y'_i$  is mapped to an original 2D caricature  $X'_i$ . The final shape is found this way:

$$X' = \sum_{j=1}^k W_{ij} X'_j$$

Once we have the final shape, the image is built using another warping algorithm.

We can now go further into 2010 to find a published paper named “*Caricature Synthesis based on Mean Value Coordinates*” [25] by Liu et. al. This paper has the particularity that tries to generate side caricatures too.

Basically this approach tries to learn from a data base the exaggeration effect to be applied to every face, and applies it later on using an interpolation method. They also bear in mind the idea of imposing likeness to the final facial composition through a measure of it.

Finally, the last paper I took into account is titled “*Rendering and Animating Expressive Caricatures*” by Obaid et. al. [26].

They are mainly the same authors as those in the third implemented paper, and it has some points in common that will be omitted as they will be detailed in the other paper.

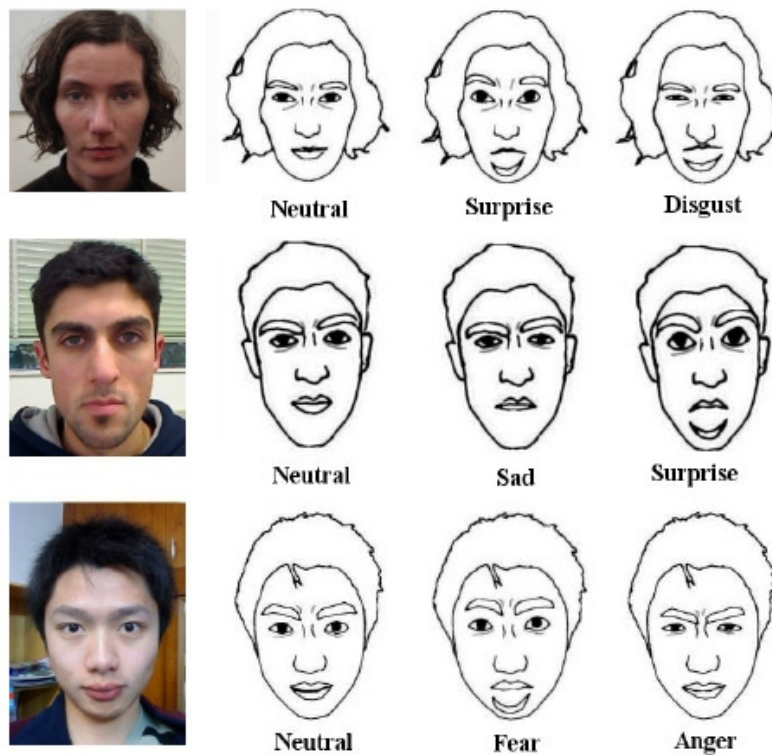
Like in the other papers, this algorithm starts by detecting the feature points; they call it the “rendering path extraction”. This is subdivided into two steps: detecting the hair and the ears, and detecting the facial components.

Detecting the hair and the ears is first done by marking the region of interest roughly and through a Canny algorithm [27] they obtain the edges, and thus, the regions.

The facial features extraction is done using AAM [18].

In order to deal with the transformation of the shape, they apply exactly the same methodology as in the third implemented paper: building Facial Deformation Tables with coefficients of rubber-sheet deformation equations.

Once this is done, their contribution is adding a bit of style to the final image by accumulating some additional strokes to the paths or lines. This is done by randomly choosing neighbor pixels to the lines and painting them in black. The results look more like a hand-made drawing:



## *First method: A Method for Deforming-driven Exaggerated Facial Animation Generation*

Before detailing the algorithm procedure, it will be easier to talk about Radial Basis Functions, in order to understand the paper better.

### - Radial Basis Functions (RBF):

Radial basis functions are an effective tool when dealing with multivariate interpolations that don't have a completely linear nature (case in which it's not recommendable to use them).

The basic nature of this kind of functions is, as the same name already indicates, completely radial, what means that regarding one function, it doesn't differentiate between all the variety of directions from the origin. It's due to the fact that these function fulfill the following:

$$\varphi(x) = \varphi(\|x\|)$$

being  $\| \cdot \|$  the Euclidean distance

More generally, letting any point be the center of the RBF:

$$\varphi(x, c) = \varphi(\|x - c\|)$$

being  $x$  any point and  $c$  the origin of the RBF

This implies that any point at a distance  $d$  from the center, independent on the vector  $dc$ , has the same value.

The basic form of a RBF is, given a set of points, and their images:

$$x_1 < x_2 < \dots < x_N, f_1, \dots, f_N$$

We suppose that these correspondences follow a rule individually like:

$$f(x_i) = f_i$$

Now, we have a function of which we only know some images and we would like to have a more general form to apply it to the rest of the space. Then, we interpolate:

$$y(x) = \sum_{i=1}^N w_i \phi(\|x - c_i\|),$$

regarding every  $x_i$  as the center of a RBF.

This is a similar form to the general interpolation functions known: it's the sum of  $N$  terms where each term provides with a certain amount to the total multiplied by a weight. We still have to find a way to determine the value of the weights. Similarly to other interpolation processes, the main constraint we have is that the points  $x_i$  must end up in the final positions (images) already given, and this will suffice.

However, unlike the other interpolation/approximation methods, the interpretation of what is really a weight and a term  $i$  is the other way around. In this case, the radial functions provide a true weight, that is, a way to indicate the influence of the term on the input value. On the other hand, the weights are the "directions" of the terms; in other words, how a point that coincides with the center must move to exactly go to the image of that center  $x_i$ .

The process of calculation of the weights clarifies this a bit more.

As it was said before, the constraint that we have is that the points must go to their images when applied the interpolation functions, so let's try to formalize it:

$$\begin{aligned} \forall i \in \{1..N\} \quad f_i = y(x_i) &= \sum_{j=1}^N w_j \varphi(\|x_i - x_j\|) = \\ &= (\varphi(\|x_i - x_1\|), \varphi(\|x_i - x_2\|), \dots, (\|x_i - x_N\|)) \cdot W^T \end{aligned}$$

This represents a row of a matrix, concretely, the  $i$ -th row. Consequently, what we see here is a matrix that we will name  $K$ .

All this turns into a final equation:

$$K \cdot W = Y \Rightarrow W = K^{-1} \cdot Y$$

where the matrix  $K$  is symmetrical, so it's invertible

This way, what we see is that in the diagonal of  $K$  there are the main differences between sources and targets, and in the adjacent cells, there's the interaction between functions, that is, how much a function affects the others when the radius of intersection of those two functions is not null.

There's still something that hasn't been defined: the radial function  $\varphi$ . In fact, there a big variety of possibilities depending on how you want the radius values to influence your system.

1. Thin-plate splines (TPS):

$$\varphi_{\text{TPS}}(x) = x^2 \log(x)$$

2. Gaussian:

$$\varphi_G(x) = e^{-x^2/\sigma^2}$$

### 3. Multiquadrics (MQ):

$$\varphi_{\text{MQ}}(x) = \sqrt{x^2 + \sigma^2}$$

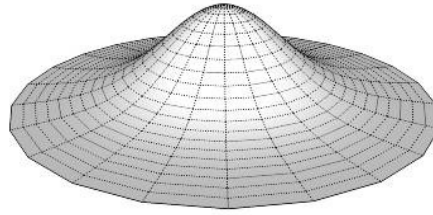
### 4. Inverse multiquadrics (MQ):

$$\varphi_{\text{IMQ}}(x) = \frac{1}{\sqrt{x^2 + \sigma^2}}$$

And some others...

All these functions have in common that, with a radius of 1 they return a value of 1 when the input point coincides with the center and they vanish as long as the distance to the center of increases.

As an example, here's the Gaussian function in 3D restricted to the radius:



#### - Explanation of the method:

I will explain this method by going through the image treatment procedure.

First of all, given an input image  $I$ , we must detect the feature points that will later be used to exaggerate the face and warp the image. This is done using the Active Contour Models (ACM) [18].

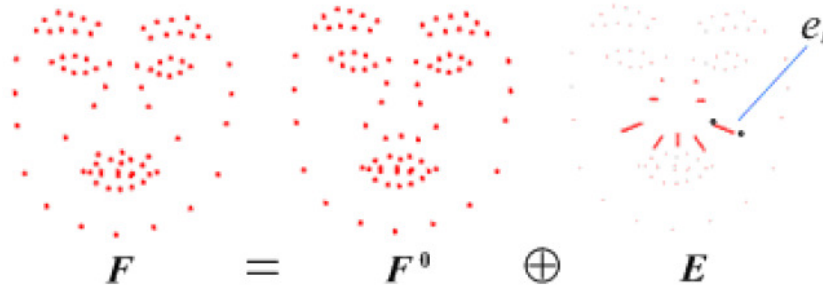
These points of the original image will be called from now on neutral face and will be denoted by  $F^0$ .

Note that every facial expression or caricature  $F$  can now be expressed as a simple addition of geometric coordinates:

$$F = F^0 \oplus E$$

being  $E$  the exaggeration/expression effect

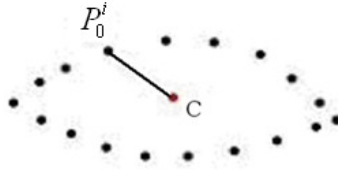
What graphically is:



As we have seen in other papers from other groups that some authors define an exaggeration effect by scaling the difference regarding a reference image and a center point for all the face or for every facial component. However, this is very limiting, as normally the exaggeration set of points is not a homogeneous scale transformation from the original one but it's formed of a set of various transformations.

To tackle this problem, the authors of this paper thought of converting the points by applying different affine transformations to every point in the set of the feature points. Restricting the set of points to a feature point like one eye, for example, we will easily see what they do:

Let's call this restricted set  $F_0$ , and  $C$  will be the geometrical center of those points. If we pick one point in the set  $P_0^i$ , we can define a feature vector as the difference between this points and the center:



This way, we can define the deformed final point  $P_i'$  as the central point plus the feature vector deformed through an affine transformation  $h$ :

$$P_i' = h(P_i^o - C) + C$$

This transformation can be decoupled into two parts, scale-rotation and translation:

$$h(\cdot) = L(\cdot) + T(\cdot)$$

What, applied to the last equation, leaves the final transformation this way:

$$P_i' = L_i(P_i^o - C) + C + T_i$$

Finding the right transformations for every point is a linear algebra problem:

$$L_i = \lambda_i \begin{bmatrix} \cos \theta_i & -\sin \theta_i \\ \sin \theta_i & \cos \theta_i \end{bmatrix}$$



where  $\lambda_i$  and  $\theta_i$  can be found:

$$\begin{cases} \theta_i = \arctan \frac{(y_i' - y_c)(x_i^o - x_c) - (y_i^o - y_c)(x_i' - x_c)}{(x_i' - x_c)(x_i^o - x_c) + (y_i' - y_c)(y_i^o - y_c)} \\ \lambda_i = \frac{y_i' - y_c}{(x_i^o - x_c) \sin \theta_i + (y_i^o - y_c) \cos \theta_i} \end{cases}$$

The authors neglect the translation transformation as they say that it is negligible in most cases.

This process can be done for all the points in the set of features points, obtaining the exaggeration effect for those points.

Up to now, we managed to get the transformed points of the desired transformation, but the initial image remains the same, so we have to apply a warping algorithm. This will be based on RBFs.

Being  $P$  a pixel in the image, and  $P'$  the desired final pixel position, the RBF interpolation goes as follows:

$$P' = P + \sum_{i=1}^n \omega_i B\left(\frac{\|P - P_i^o\|}{R_i}\right)$$

The RBF chosen is detailed in [20].

As you can see, there's a parameter that we didn't define while describing how the RBFs work; it's  $R_i$ . This is the radius of influence of every point  $i$ . Considering that the default radius of the functions is 1 in addition to the fact that the minimum Euclidean distance between two pixels is exactly 1, if we leave the radius this way, only the feature points will be transformed. In the paper it's not explained how they defined those radius, so it may be manual work according to the taste of the artist.

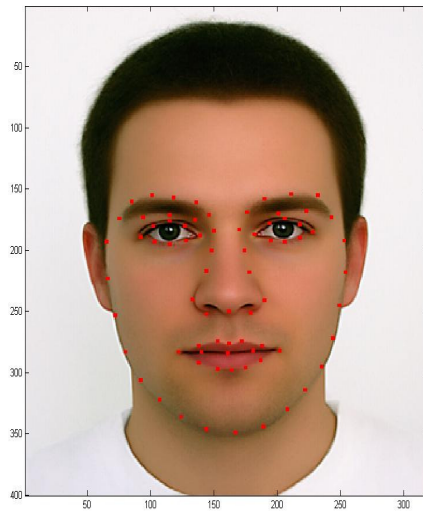
#### - Implementation and results:

All the code I did is in Matlab 7.6.0 because it's very flexible and it provides an easy way to visualize the results in an image (instruction: `image(·)`).

The algorithm is run with a jpg image found in Internet:



First of all is finding the feature points of the image through ACM. I won't go into detail on this algorithm as feature detection is not the topic of this thesis.

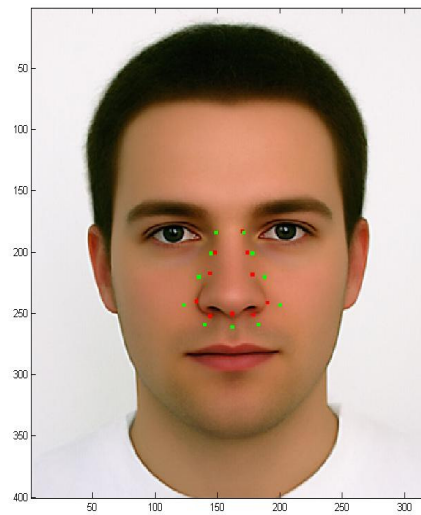


Now we must find the central point of the facial component by calculating the geometrical mean of the original points.

Bearing in mind that this is a tool for the caricaturists, there's no rule to determine what features must be exaggerated nor in what way.

I will focus on the nose to go through the process.

As an amateur caricaturist, I spend some time finding acceptable scaling factors and angles with which to create the affine transformation that will be applied to every point in the nose and here is the result:



Everything is going to be applied to the feature vectors, so now these must be found to go further.

Right after that, we must build the structure with which to run the RBFs on the image, that is, finding the weights.

In this case, the function that must be interpolated goes from a 2D interval into another 2D interval, therefore, the weights are going to be 2D as well.

The matrix is found by applying the theory given previously.

```
function [weights]=findRBFWeightsDirections(fPoints,eDirections,radius)

for i=1:length(fPoints)
    for j=1:length(fPoints)
        if(i==j)
            matrix(i,j)=1;
        else
            norma=norm(fPoints{i}-fPoints{j})/radius(i);
            if(norma>1)
                norma=1;
            end
            matrix(i,j)=fRBF(norma);
        end
    end
end

invMatrix=inv(matrix);
```

But the  $Y$  vector doesn't correspond to the images of the points. As it's seen in the definition of the RBF interpolation, the function domain of every RBF is not formed with the pixel positions, but with the feature vectors; therefore, the image set must be the group of the image points subtracting them the center point of the facial feature too. In practical points, we get that  $Y=\{(x_i, y_i - x_i)\}$ .

With this, we can eventually calculate the interpolation weights:

```
for i=1:length(fPoints)
    weights{i}=[0,0];
    for j=1:length(fPoints)
        weights{i}=weights{i}+invMatrix(i,j)*eDirections{j};
    end
end
```

end

Finally, we only have to check the results applying these functions to the whole image:

```
function [imatgeOut]=warpedImageIntermediateDirections(imatge,fPoints,weights,radius)

n=size(imatge);

imatgeOut=imatge(:,:,:)';
imatgeOut(:,:,:) = 0;

for i=1:0.5:n(1)
    for j=1:0.5:n(2);
        posPixel=[i,j];
        for k=1:length(fPoints)
            a(1)=i-fPoints{k}(1);
            a(2)=j-fPoints{k}(2);
            norma=norm(a)/radius(k);
            if(norma>1)
                norma=1;
            end
            posPixel=posPixel+weights{k}*fcosRBF(norma);
        end
        imatgeOut(floor(posPixel(1)),floor(posPixel(2)),:)=imatge(floor(i),floor(j),:);
    end
end

image(imatgeOut);

end
```

The resulting image of applying the RBFs to the pixels would be an image with some black points, so I also applied the functions to the intermediate pixels to fill in the gaps.

Here is the final image:



- Advantages of the method:

This method offers the possibility of warping images in a very nice and smooth way from right initial deformations of the feature points.

This paper, even though they don't seem to do it, a learning phase where the possible exaggeration effects can be preconstructed and offered to the user as possible patterns to follow.

It is a quite flexible approach with which to generate caricatures, as the affine transformations that must be defined don't have any limitation.

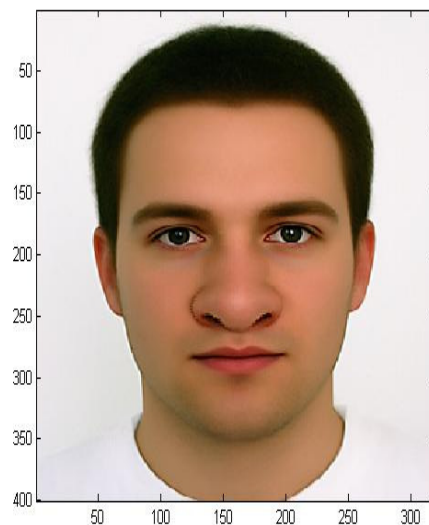
- Drawbacks and complaints:

While implementing this paper I have seen that some of the points were not explained clearly, and it was you who had to make out what they had done there, even though it is not a serious complaint, as it is a habitual feature in the papers published unfortunately.

I will order the problems I found in the paper according to its procedure, so let's start with the features detection.

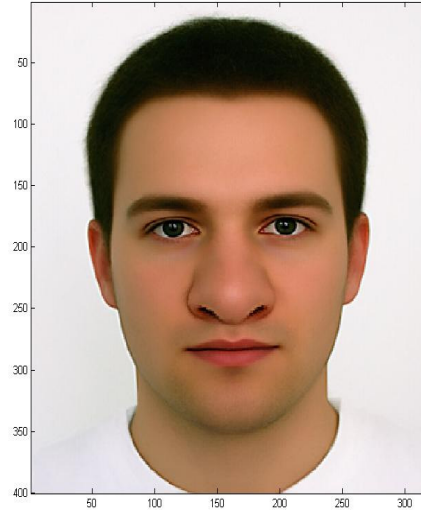
The authors chose ACM to detect the features of the face. With this algorithm they obtain a set of points for every facial component. Apart from that, they chose RBFs to do the warping task on the image. These two algorithms don't have to work well together, as there are lots of points with which you can describe a facial feature and the selected ones don't necessarily have to be the optimal ones to produce nice and smooth pixel movements. If the nature of the interpolation function had been homogeneous, one set or another wouldn't make a difference. However, as it can readily be seen, RBFs are not linear in any way, so the set of points  $\{x_i\}$  must be selected accurately bearing in mind the kind of movement will be carried out. And it's not only that what's a bit critical, but also the amount of points given because it may be necessary to have an additional point to be able to describe a complex movement.

I tried myself to change the points to see what happened, and the results were a complete disaster. The radius of influence also had something to say about it, and it was not even mentioned in the paper how they specified those values. After changing only 2 points, the results are the following:



It is even clearer than in the final result of the implementation that in the upper part of the nose there has been a radial movement applied, and between the upper part and the down part of the nose very little changes have been done, thus, this nose looks weird.

On the other hand, I had another try and moved 6 points considering that the deformation was going to look much more natural, and here is the result:



If a method is going to be a tool for caricaturists, it can't be so rigid in something so easily modifiable.

While I was implementing the function to calculate the weights I found that the method that they suggest to find a good angle and scale factor didn't work. It gave very strange results that had nothing to do with what I expected. In order to deal with it, I substituted the two formulas for a very simple pair:

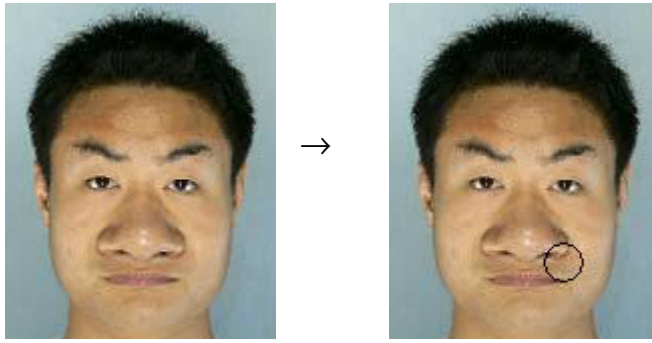
Being  $v = P_i^0 - C, v' = P_i' - C$  :

$$\theta = a \cos(v \cdot v' / \|v\| \cdot \|v'\|)$$

$$\lambda = v' / v$$

The angle doesn't have to be oriented, so what I did was interpreting the vectors  $v$  and  $v'$  as in 3D (adding a third 0 component) and did the cross product of them. If the resulting  $z$  component was positive, it meant that the rotation was counter-clockwise (positive), otherwise, it was clockwise (negative).

Another problem found was that the image was warped, what meant that the shadows in the original image go with the points, with no recalculation. Down the nose, you will see how the shadow is wider than how it should be.



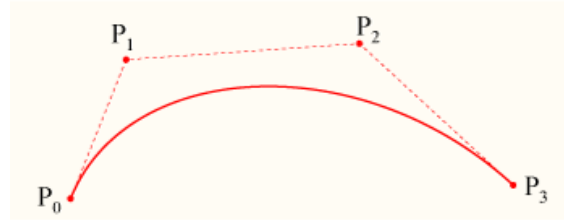
*Picture taken from the paper*

## *Second method: Exaggeration of Facial Features in Caricaturing*

In this paper, the main tool used are Bézier curves. This approach is classified in the group of those that let the caricaturist work freely, offering the possibility to computerize the drawing. So as to take in the paper better, I will talk about the Bézier curves first, and later on we will see its application to the paper.

### - Bézier curves:

Bézier curves are functions that look this way



and must fulfill some rules:

Given a set of  $p_0, \dots, p_n$  points

1. The curve must pass through the first and the last point:  
 $C(0) = p_0, C(1) = p_n$
2. The curve is completely contained in the convex hull of the control points
3. Translations and rotation applied to the curve must leave it as if the transformation had been solely applied to the control points.
4. The basic form of a Bézier curve is this way:

$$C(x) = \sum_{i=0}^n p_i \cdot B_{in}(x)$$

where  $n$  is the number of control points +1

$p_i$  is a control point

$B_{in}(x)$  are the Bernstein polynomials

$x \in [0,1]$

The Bernstein polynomials are the following:

$$B_{in} = \binom{n}{i} t^i (1-t)^{n-i}$$

A practical way of understanding what a Bézier curve is may be through imagining that you have a straight line and you add some points that pull the line without touching it and



keeping the constraint that the curve must be in the convex hull of the set of control points.

- Explanation of the method:

As I said before, this method is aimed to help the caricaturists build their drawings more freely. The tools that they will be given are Bézier curves.

Like in almost all the papers, they must detect the features of the face first to anything. In this case, following with the nature of this group of algorithms, the detection is completely manual.

Once this is done, it's the task of the artist to decide what parts to exaggerate while keeping the overall look of the face.

In this paper, they offer a list of rules with which one can easily exaggerate facial features once detected. For instance, directly from the paper:

**Rules for the Nose**

There are several characteristics of our nose, such as size and shape, etc. There are three rules for the nose:

Rule 1: If the nose is big, we enlarge the size of the nose by using the bottom of the nose as the central point.

Enlarging the size may cause the nose to cover part of the mouth.

Rule 2: If the nose is hook-nosed, we make the nose tip lower and keep the other parts of nose the same. On the contrary, if the nose is snub-nosed, we make the nose tip higher.

Rule 3: If the nose is high-bridged, we can bend the bridge of the nose to emphasize this characteristic.

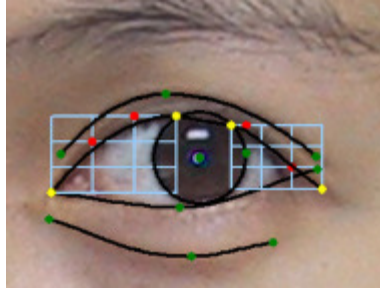
This method has the particularity that the input images must be rotated to the side about forty-five degrees in order to see the exaggeration effect a bit better.

Every facial feature will then be associated a set of Bézier curves. In fact, every pair of two neighbor points will have their own Bézier curve to describe the curvature. However, having the starting and end points is not enough to define a Bézier curve, we need more control points, otherwise, the curves defined will always be straight lines.

The authors indicate that they choose two more control points to define the curve. Then, all the Bézier curves that we will be working with will be cubic, and will have this form:

$$\mathbf{B}(t) = (1 - t)^3 \mathbf{P}_0 + 3(1 - t)^2 t \mathbf{P}_1 + 3(1 - t) t^2 \mathbf{P}_2 + t^3 \mathbf{P}_3, \quad t \in [0, 1].$$

They add that the placement of the control points will be in a range of a 3x3 matrix between the points.



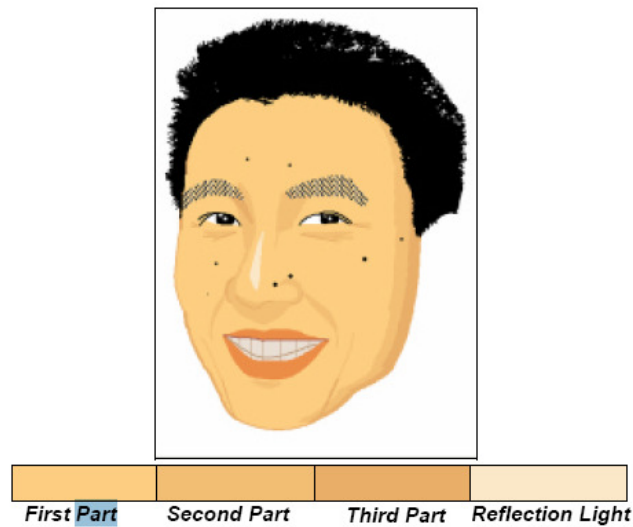
Nevertheless, they don't give any clue on how to build them, so it's again manual work of the artist to decide on the curvature of every part of the face.

In this paper, it's not only line drawing what they do, but they also add the hair of the original picture and they apply a tone-shading look to the caricature.

The hair is found indicating two starting pixels in the original image and running a Bread-first Search algorithm to find the rest of the hair. Once it's done, they store it separately.

As for the tone-shading, they divide the colors into 4 tones:

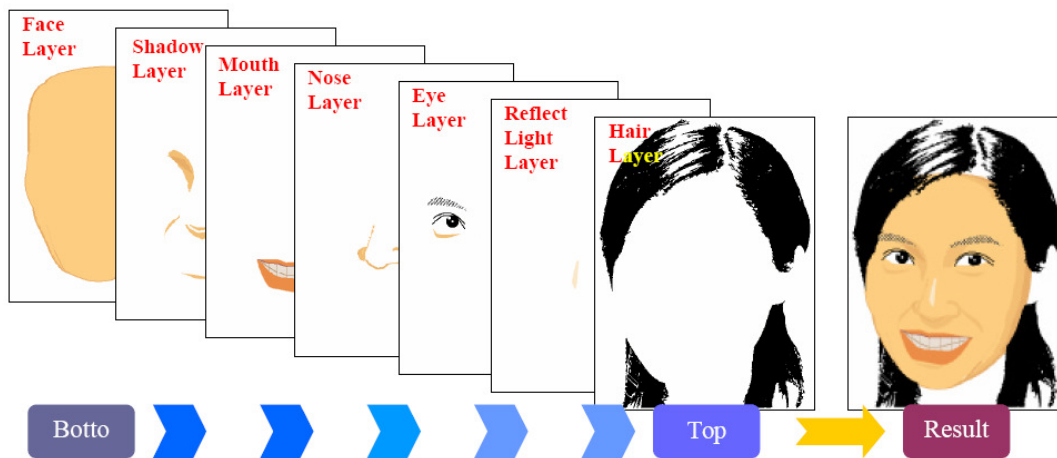
The first one is ninety percent white, five percent yellow and five percent orange  
 The second and third parts are heavier in orange and yellow  
 The forth part has more white



They loosely indicate where the shadows should be so that it's easier for the user to paint them.

Therefore, briefly, the procedure of this paper is to manually choose the landmarks, choose the regions to exaggerate and in what way, getting the hair, assemble all the parts together with the hair and paint it meanwhile.

The authors also took something else into account: once the features are individually exaggerated, they may intersect one another, so they built a layer rendering method so as to tackle the problem. First of all comes the face, then the shadows of the face, then the mouth, after it comes the nose, then the eyes, the reflected lights and the hair.



Once everything is done, we get the final caricature.

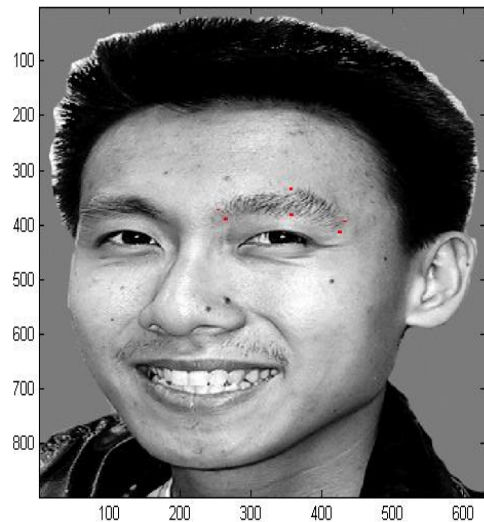
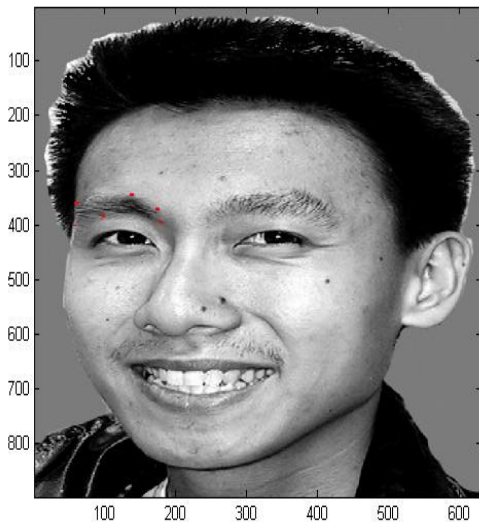
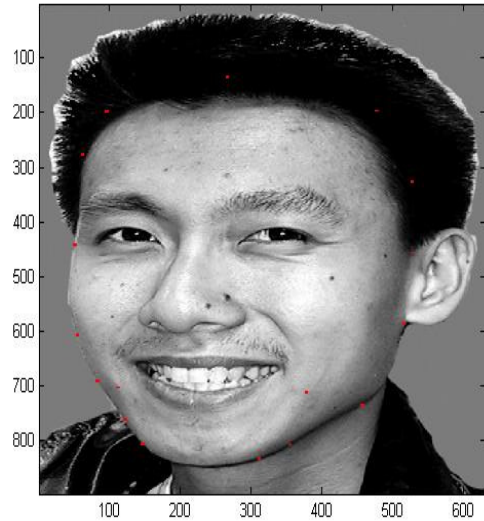
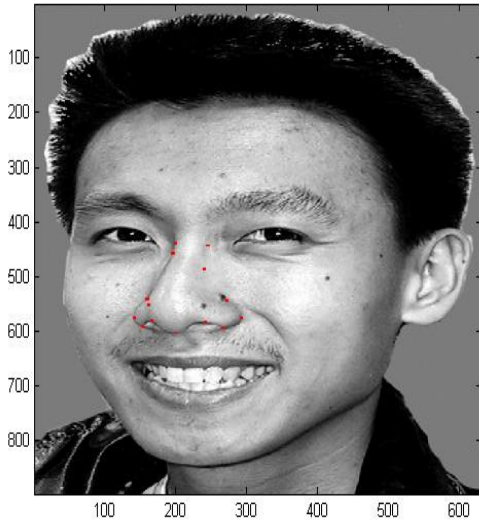
- Implementation and results:

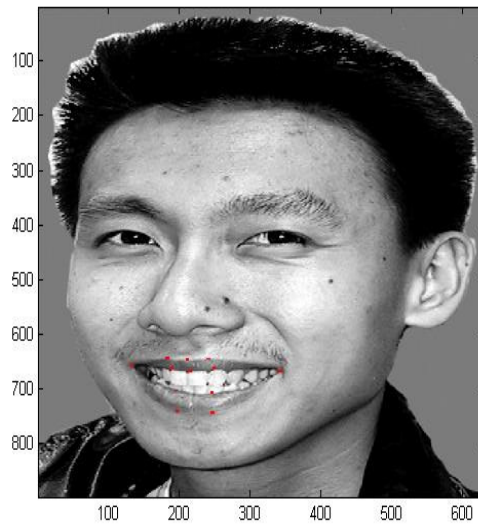
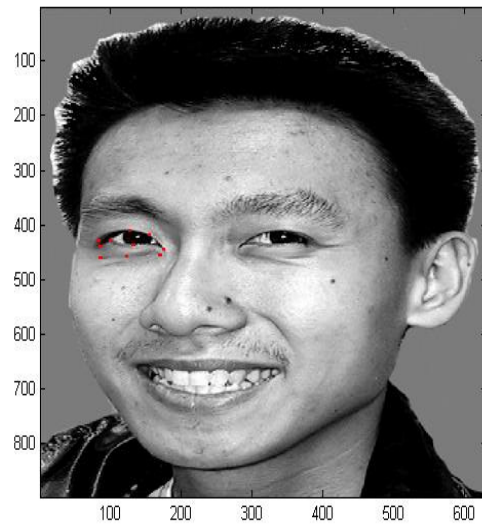
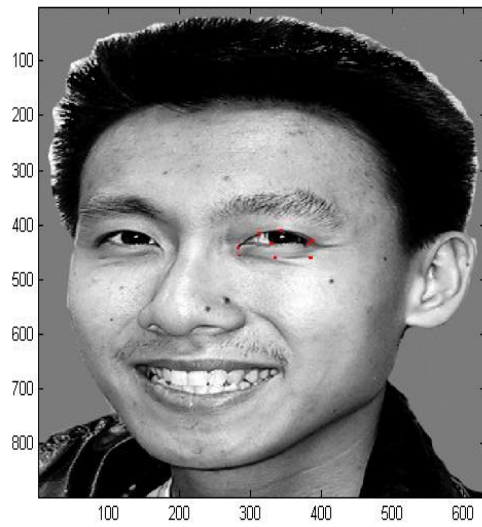
All the functions to implement this paper were also done with Matlab, as it provides the function `image()`, to see the results immediately. The picture used to create the caricature was the original one from the paper in order to make it easier to compare my results with theirs.



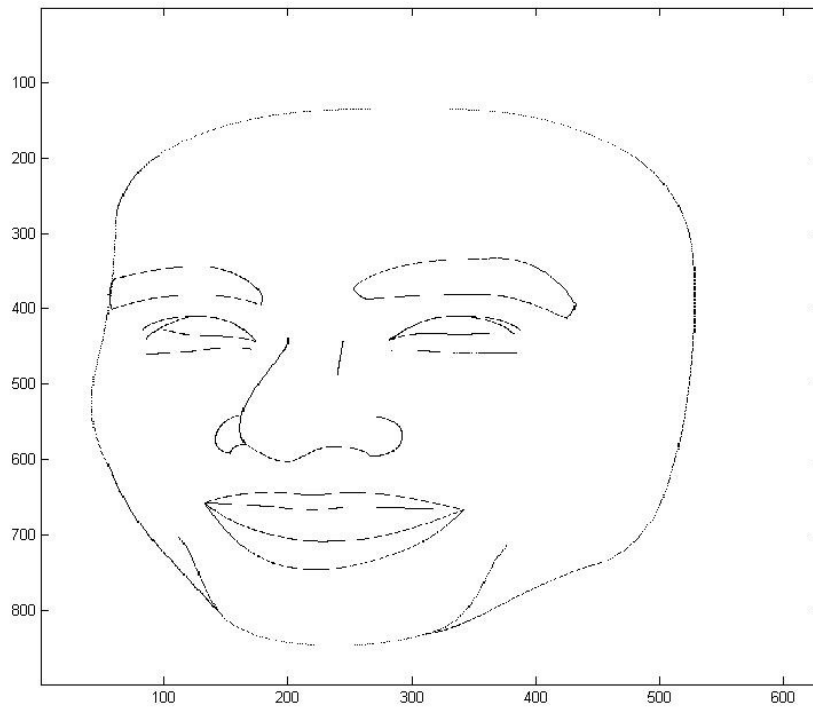
Implementing this paper became a test to patience. All that had to be done was placing a large amounts of points to detect the facial features and then, place more points to describe the Bézier curves trying to fulfill with derivability between neighbors at the same time.

The feature landmarks (the red spots) that I put where the following:





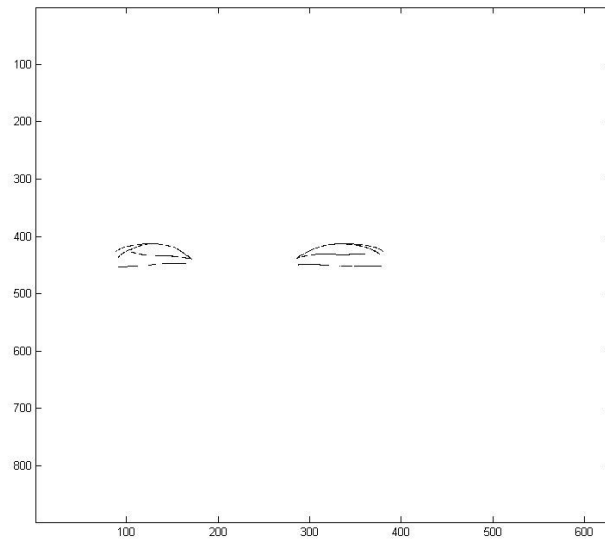
Once it was done, I set the control points for all the Bézier curves and the resulting image is the one below:



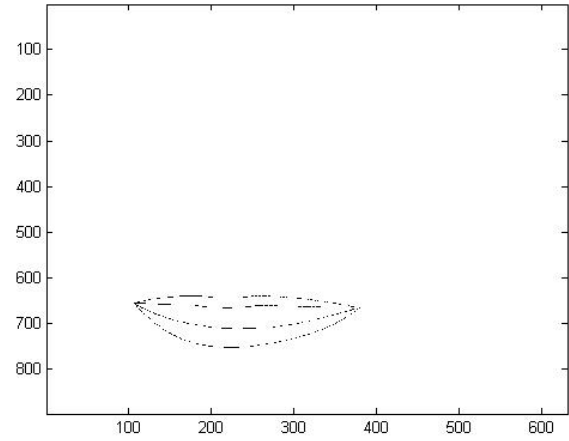
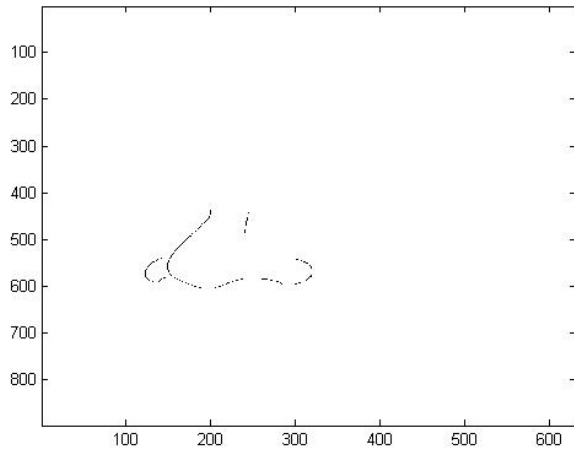
As it's indicated, now we have to choose what features you want to exaggerate.

I found that the eyes were very small, so I shrunk them considering the center of the eyes and shrinking the difference by  $(\lambda_x, \lambda_y)$ .

In this case I chose a 10% less in x and a 20% less in the y:

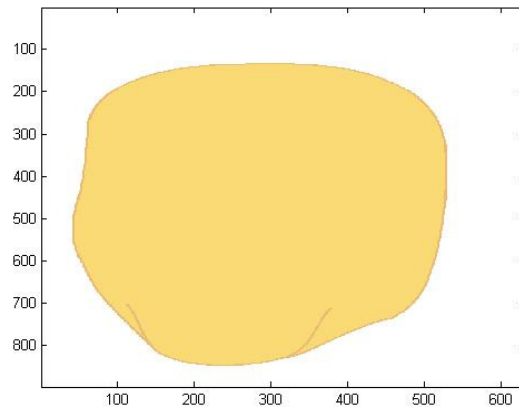


After that, I thought that the mouth was very big, so was the nose. Therefore, I enlarged them:

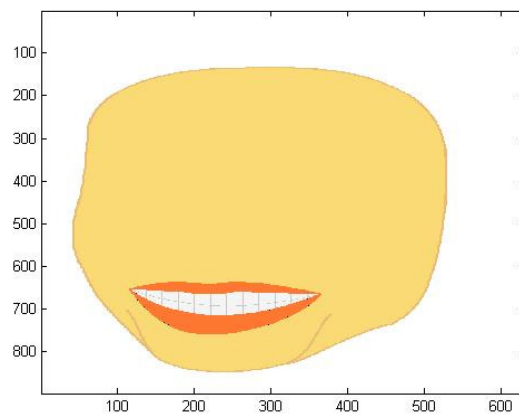


I stored all the transformed feature points and control points, and started with the layer rendering part of the method.

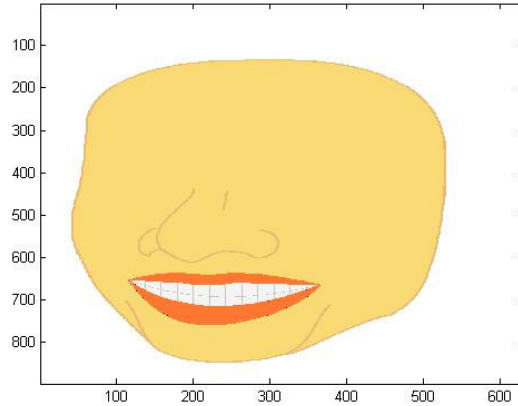
First of all, the face with the base color:



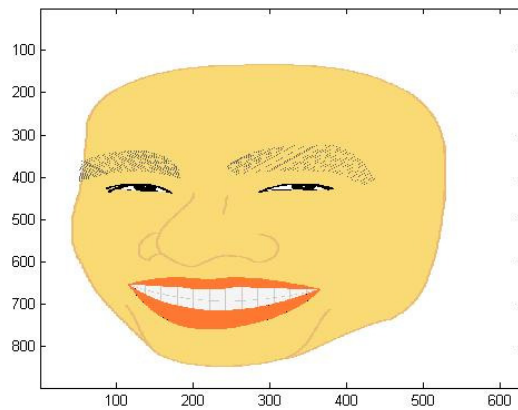
Now comes the mouth:



And after it goes the nose:



What is followed by the eyes and the eyebrows:



Finally, the hair and the shadows to obtain the final image, adding the spots to the skin as the authors do. The shadows were built creating other Bézier curves limiting the areas and painting in there.





- Advantages of the method:

The main advantage of this method is that it's completely flexible. Every single point can be defined anywhere, every single curve can be determined in any way.

Moreover, I must add that the results, after having been very patient were very good. The caricature was funny, it still kept a resemblance to the original photo and aesthetically it is very nice. If the process of creation wouldn't be so tedious, it would be a serious approach to bear in mind.

- Drawbacks and complaints:

As I already mentioned in the implementation process, it was very tedious work. The fact that it's very flexible, to my opinion, is not compensated with the amount of days you end up spending setting points in an image so that the curves look well enough.

However, even though this method flees your creativity, there were still some small limitations in the process. The authors imposed using cubic Bézier curves, what sometimes meant not finding the right curvature for a pair of points. Moreover, they suggest a number of points per facial component, what, depending on the case, may not be sufficient, and this is translated into more tedious work done.

### *Third method: Facial Caricature Generation Using a Quadratic Deformation Model*

Again, before deepening in this paper, we must have some background: rubber-sheet transformations [28] and FACS [29].

#### - Rubber-sheet transformations:

##### *a) Definition:*

Rubber-sheet transformations are defined by a set of polynomials of degree  $n$ . In practice, they are used to describe extensive areas that could be covered with something like a blanket leaving no gap. These areas can be of any dimension.

In this paper, what we need is the 2-dimensional expression of the rubber-sheet equations:

$$\begin{aligned}x'_i &= a_1 x_i^2 + a_2 x_i y_i + a_3 y_i^2 + a_4 x_i + a_5 y_i + a_6 \\y'_i &= b_1 x_i^2 + b_2 x_i y_i + b_3 y_i^2 + b_4 x_i + b_5 y_i + b_6 \\i &= 1, \dots, n\end{aligned}$$

Then, all that needs to be done is, from a set of input and output data, finding the coefficients of the polynomials.

##### *b) Finding the coefficients:*

As it can readily be seen, we need at least 12 correspondences between  $x$  and  $x'$ , and  $y$  and  $y'$  in order to have enough information to find a determined system with a unique solution.

Following the principles of least-squares errors, we will try to form a system of equations from the original ones, the solution of which will also be a solution to the rubber-sheet equations.

Here are the equations of the sum of the differences squared; what gives an idea of the total error for a set of coefficients:

$$\begin{aligned}G_x &= \sum [x'_i - (a_1 x_i^2 + a_2 x_i y_i + a_3 y_i^2 + a_4 x_i + a_5 y_i + a_6)]^2 \\G_y &= \sum [y'_i - (b_1 x_i^2 + b_2 x_i y_i + b_3 y_i^2 + b_4 x_i + b_5 y_i + b_6)]^2\end{aligned}$$

As an error function, this has to be minimized with the purpose of achieving good approximation of the inherent transformation between the sets of points  $\{x_i\}$  and  $\{y_i\}$ . This can be done by not letting it grow:

$$\frac{\partial G_x}{\partial a_j} = 0 ; \quad \frac{\partial G_y}{\partial b_j} = 0 \quad j = 1, \dots, 6$$

This seems to be a lot of calculation, although if we develop one of the derivatives, for instance, the derivative with respect to  $a_1$  like they do in the paper, we will see that it will end up being a straight forward process.

Direct derivation:

$$\frac{\partial G_x}{\partial a_1} = 0$$

$$2 \sum_{i=1}^n \left[ x_i' - (a_1 x_i^2 + a_2 x_i y_i + a_3 y_i^2 + a_4 x_i + a_5 y_i + a_6) \right] x_i^2 = 0$$

Putting inside the sum the multiplying term of  $a_1$  ( $x_i^2$ ):

$$\sum_{i=1}^n \left[ x_i' x_i^2 - a_1 x_i^4 - a_2 x_i^3 y_i - a_3 x_i^2 y_i^2 + a_4 x_i^3 - a_5 x_i^2 y_i - a_6 x_i^2 \right] = 0$$

Separating the individual sums for every coefficient:

$$\sum_{i=1}^n x_i' x_i^2 - \sum_{i=1}^n a_1 x_i^4 - \sum_{i=1}^n a_2 x_i^3 y_i - \sum_{i=1}^n a_3 x_i^2 y_i^2 - \sum_{i=1}^n a_4 x_i^3 - \sum_{i=1}^n a_5 x_i^2 y_i - \sum_{i=1}^n a_6 x_i^2 = 0$$

And here we can see the first term minus two vectors multiplied, one with the  $a_i$  coefficients and the other with the resulting summation.

$$\sum_1^n x_i' x_i^2 - \left( \sum_1^n x_i^4, \sum_1^n x_i^3 y_i, \dots, \sum_1^n x_i^2 \right) \cdot \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ \dots \\ a_6 \end{pmatrix} = 0$$

This gives the first row of a matrix. Then, doing the same process for the rest of the coefficients, we end up with a matrix M, what is simply getting the initial summations for every coefficient and multiplying it by the derived coefficient term.

Once the matrix of m coefficients has been found, we still have the equation of the  $a_i$  to solve.

$$\begin{pmatrix} \sum_1^n x_i' x_i^2 \\ \dots \\ \sum_1^n x_i' \end{pmatrix} - \begin{pmatrix} \sum_1^n x_i^4 & \dots & \sum_1^n x_i^2 \\ \dots & \dots & \dots \\ \sum_1^n x_i^2 & \dots & 1 \end{pmatrix} \cdot \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ \dots \\ a_6 \end{pmatrix} = 0$$

Naming  $N$  the first vector,  $M$  the matrix and  $P$  the vector of the  $a_i$  coefficients, we obtain:

$$N - M \cdot P = 0 \Rightarrow N = M \cdot P \Rightarrow P = M^{-1} \cdot N$$

And thus, we have the  $a_i$  coefficients of the rubber-sheet equations. Doing the same for the  $y$  values we will obtain the  $b_i$  coefficients.

Note that  $n$  is the number of samples that we have, and it's not only limited to 6, but it allows to give as much information as necessary to define the polynomial accurately.

#### - Facial Action Coding System or FACS:

A human face has lots of muscles to control the facial movements. Movements of these, together with other muscle movements form our facial expressions. The aim of this technique is to codify the expressions that people perform in terms of the muscles necessary to do them.

Some muscles don't usually work alone when forming expressions, and that's why Ekman grouped them in Action Units (AU), what offers a way to easily codify all the expressions by making combinations of the 64 available ones.

Here is a list of examples of the facial groups of muscles (AU) that Ekman found:

1. Inner Brow Raiser (*Frontalis, pars medialis*) (AU=1)



2. Brow Lowerer (*Corrugator supercilii, Depressor supercilii*) (AU=4)



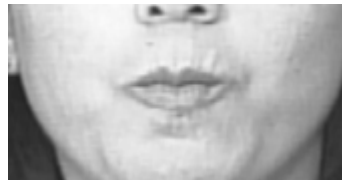
3. Upper Lip Raiser (*Levator labii superioris*) (AU=10)



4. Lip Corner Depressor (*Depressor anguli oris* (a.k.a. *Triangularis*)) (AU=15)



5. Lip Tightener (*Orbicularis oris*) (AU=23)



We could get as an example the sad face (the AUs of which will be used to create the facial expression in the implementation part of the algorithm).

SAD : 1+4+6+11+15+17

1, 4 and 15 AUs are listed below, the other Aus are these:

6: Cheek Raiser :



11: Nasolabial Deepener :



17: Chin Raiser :



All these Aus together form a universal sad face:



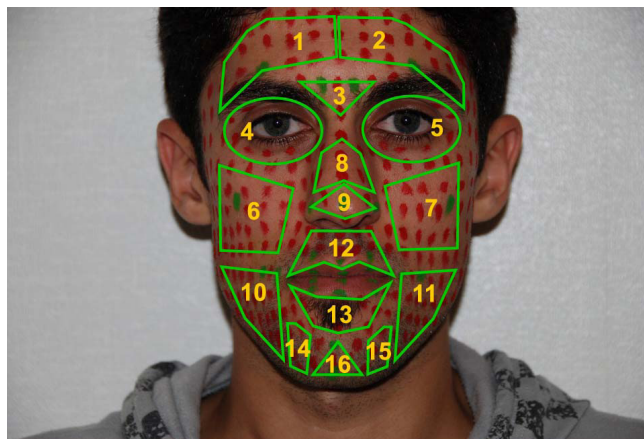
Picture of Eve (Paul Ekman's daughter)

Before talking about the paper, we also need to talk about a previous paper done by the same authors, which is one of the key features of the third paper. It's "*Facial Expression Representation Using a Quadratic Deformation Model*" from 2009 [30].

- Pre-paper: Facial Expression Representation Using a Quadratic Deformation Model:

The main contribution of this paper is the way of describing the six universal facial expressions according to Ekman (happiness, sadness, anger, surprise, fear and disgust) through rubber-sheet transformations. As these expressions are described in a universal manner, they understand this method to be general. Once the rubber-sheet transformations are built, they will be applied in the following paper to other input images.

Those rubber-sheet transformations are not fine enough to code all the facial movements with only two equation of degree 2, so the authors first divided the face into various facial regions which will be applied a rubber-sheet to, 16 in total, as it can be seen in the picture below:



After that, every facial region for every facial expression will be associated to a rubber-sheet transformation.

In order to get the transformation, they must have data to work with first. The authors hire some actors and ask them to perform the six universal expressions in addition to the

neutral face. The expressions are captured making use of markers on the face and a tracker.

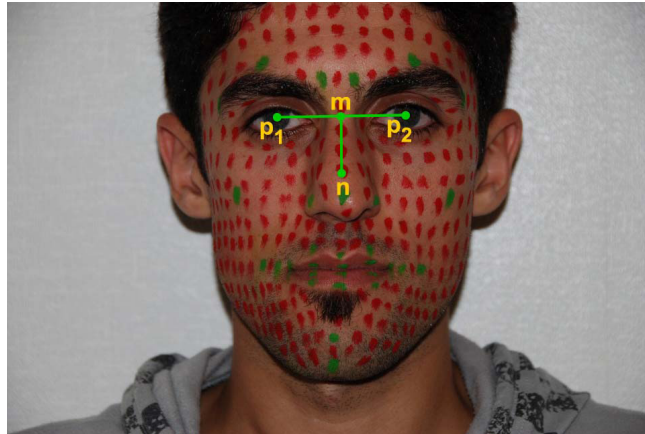
After acquiring the data from the markers for every facial expression, they subdivide them into the sixteen regions already mentioned.

Now, we have data to which the rubber-sheet transformations can be applied.

Every facial region will for one rubber-sheet, being the neutral face the original input and the expressive face the target one.

Nevertheless, if we found coefficients for these regions, they would never be general, as they are subject to the image, rotation and scaling factor, thus, a normalization process should come first.

The normalization process they follow is a very standard one. First of all, we must find a reference point in the face, some parts of the face that, independent on the image, they are going to be fixed or they are going to give a scaling factor with which to modify the image. The authors chose the eyes iris and the geometrical center between both.



Point  $m$  is going to be the origin, and  $p_1$  and  $p_2$  are going to be used to do the rotation and the scaling on the image.

The first thing to do is subtracting the origin  $m$  to the whole image. Now we need to find the angle that form the vector  $p_1p_2$  with a horizontal line ( $[1\ 0]$  vector) and the scaling factor to end up with the coordinates  $(1\ 0)$  for  $p_2$ , and  $(-1\ 0)$  for  $p_1$ . This was done like in the first algorithm implementation:

Being  $v = P_i^0 - C, v' = P_i' - C$  :

$$\theta = \arccos(v \cdot v' / (\|v\| \cdot \|v'\|))$$

$$\lambda = \|v'\| / \|v\|$$

They rotate and scale the images once these values are found.

Now, we do can find the coefficients of the rubber-sheet transformations to the images as they are going to be on a normalized face, the way it has been indicated in the explanation of them above.



After that, we have a bunch of coefficients: 12 (# coefficients a single rubber-sheet transformation has) times 16 (the number of regions in a face) times 7 (the number of expressions and the neutral face) times the number of actors hired.

The authors decided then to form a unique table called the “Facial Deformation Table” (FDT). This will be a total of six tables, as many as expressions, where every table will hold the average coefficients of every region stored in order to achieve that certain facial expression.

An example of the table for the smile face is this one:

Deformation Parameters													
	a <sub>1</sub>	a <sub>2</sub>	a <sub>3</sub>	a <sub>4</sub>	a <sub>5</sub>	a <sub>6</sub>	b <sub>1</sub>	b <sub>2</sub>	b <sub>3</sub>	b <sub>4</sub>	b <sub>5</sub>	b <sub>6</sub>	
Facial Region Number	1	-0.003	0.009	-0.002	0.991	0.007	0.000	-0.007	0.015	0.011	-0.018	0.987	0.000
	2	-0.002	0.000	-0.007	1.005	0.015	0.000	0.005	0.005	0.004	-0.021	0.998	0.000
	3	-0.016	-0.119	0.093	1.070	-0.045	-0.001	0.009	-0.037	0.073	-0.009	0.951	0.001
	4	-0.009	-0.019	-0.104	0.977	-0.024	0.000	-0.009	0.086	0.101	-0.015	0.994	0.001
	5	0.001	-0.005	0.104	1.001	0.026	-0.001	-0.012	-0.126	0.089	0.021	1.037	0.001
	6	0.040	-0.041	-0.026	1.054	-0.012	0.000	-0.078	0.022	0.003	-0.194	1.012	0.000
	7	-0.050	-0.088	-0.031	1.035	-0.048	0.000	-0.082	-0.030	-0.015	0.197	0.992	0.000
	8	0.064	-0.035	-0.022	0.978	-0.023	-0.001	0.132	-0.021	0.057	-0.022	1.072	0.004
	9	-0.024	-0.207	0.010	0.803	0.008	0.000	0.247	-0.059	0.010	-0.088	1.048	0.000
	10	0.157	-0.007	0.022	1.387	-0.012	0.000	-0.113	-0.028	-0.020	-0.286	0.931	0.000
	11	-0.107	-0.008	-0.048	1.222	-0.112	0.000	-0.132	0.022	-0.028	0.314	0.911	0.000
	12	0.013	0.050	-0.006	1.308	-0.012	0.000	0.107	0.018	0.007	0.047	0.961	0.000
	13	0.021	0.065	0.005	1.326	0.015	0.000	0.306	0.057	-0.023	0.133	0.972	0.000
	14	0.318	0.061	0.003	1.609	-0.014	0.000	-0.173	-0.322	0.073	-1.521	1.315	0.000
	15	-0.321	0.066	0.003	1.673	0.042	0.000	-0.218	0.146	0.038	1.052	1.222	0.000
	16	0.044	0.153	0.005	1.632	0.015	0.000	0.003	0.036	0.023	0.042	1.110	0.000

- Explanation of the method:

In this paper, what the authors do is transferring the Facial Deformation Tables they created in the previous paper to other images.

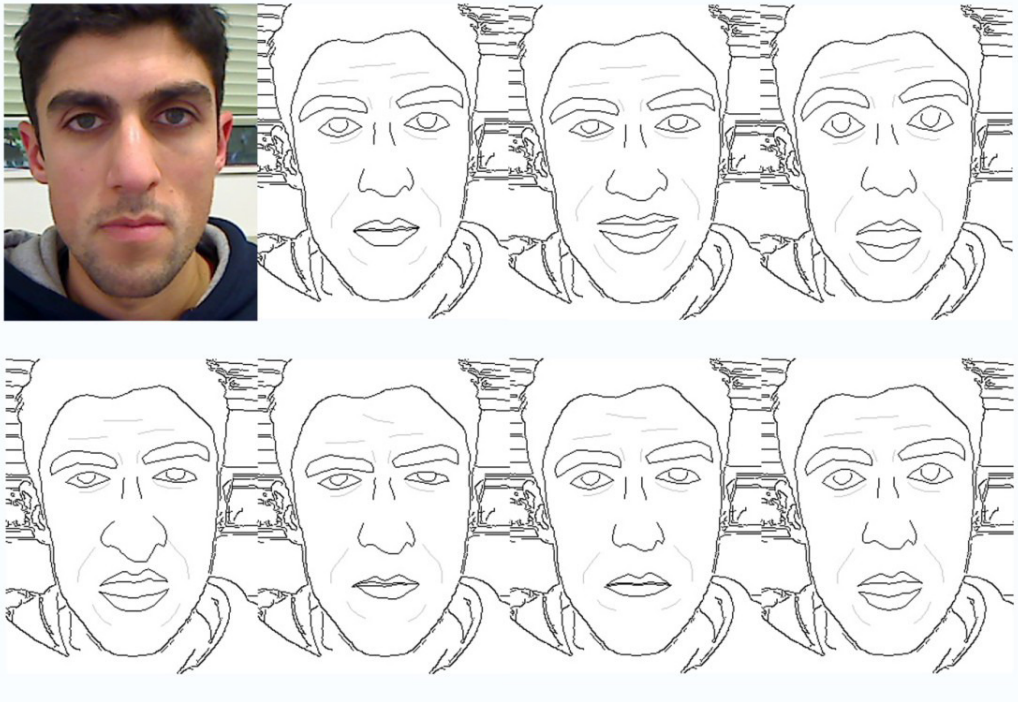
The process goes as follows:

Similarly to the other methods, they detect the facial features first. In this case, they use AAM [18]. Moreover, they detect the hair, the ears and the background. Afterwards, they organize the features in regions corresponding to the regions mentioned in the previous paper.

Now, the authors normalize the image and make use of the deformation tables they built in the other paper.

Once the facial features have been deformed towards a certain facial expressions, they simply draw the lines and get the final caricature by also assembling it to the background lines.

Here are the results:



- Implementation and results:

This paper was also implemented in Matlab 7.6 for the same reason as in the other papers.

The image that will be used will be the same one as in the first implemented paper:



As I will explain later in the subchapter “Drawbacks and Complaints”, I don’t agree in the way they captured the expressions of a person. Not even partially do they capture the idea of what FACS is, and the results they give show it clearly, as the images obtained are represent vaguely at most every facial expression, if not they represent another expression.

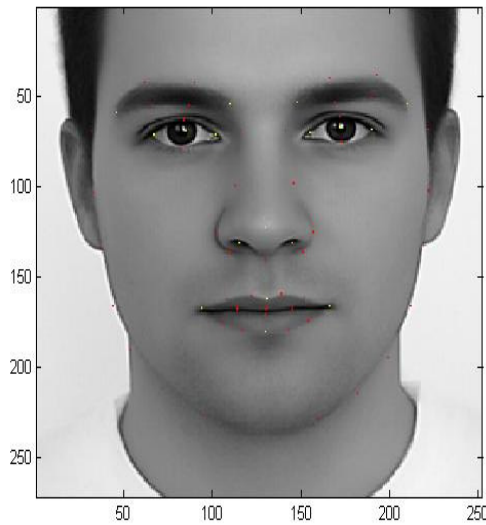
Due to that, in addition to the fact that I work with facial expressions habitually, I decided to improve this fact and see if the results were better.

I grabbed every Action Unit of a facial expression and found the appropriate transformation of all the points in the lattice I put on the image (I will show you the lattice later on). Basically, what I did was making the most of the main idea of the paper, what was to transform the expressions with rubber-sheet transformations, keeping as much as possible the essence of FACS.

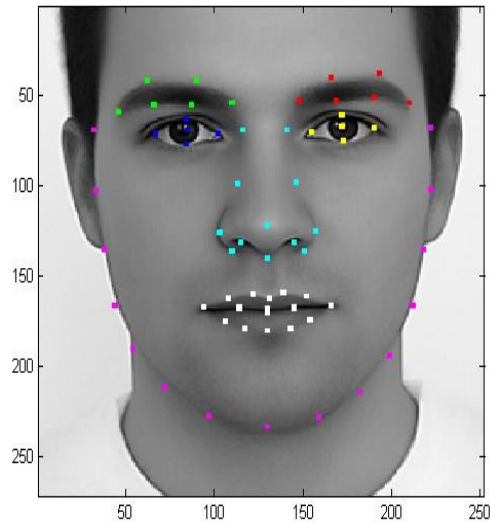
The final procedure is almost the same as the one in the paper though not extracting the expressions from actors and applying an average on the coefficients (explained later), but building every transformation myself being very accurate with the description of every AU forming a facial expression.

Once I achieved that, the algorithm remains the same.

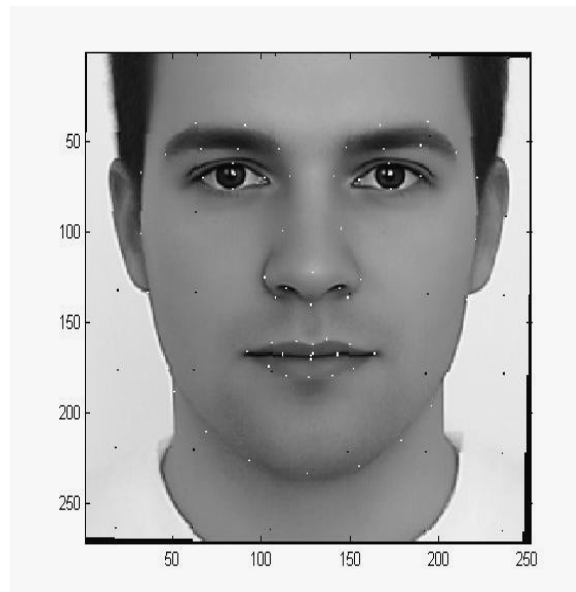
First of all, we must get the feature points:



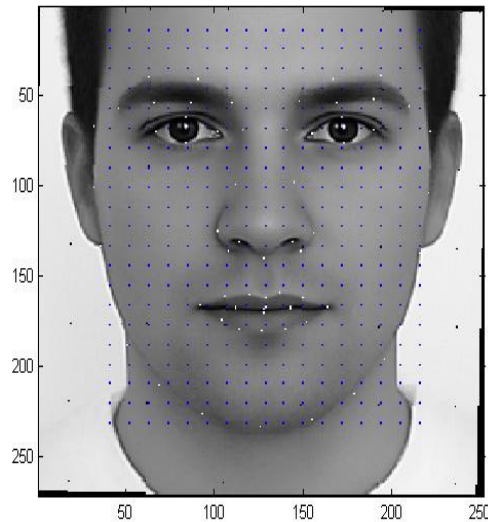
Now, I split the points into regions:



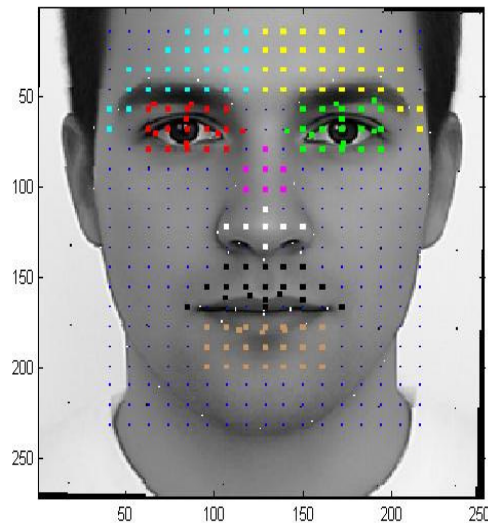
Once we have this, I normalized the image from the center of both eyes and the mid point between them ( $p_1$ ,  $p_2$  and  $m$  respectively in the paper).



Now I applied a lattice that emulated the amount of markers that will be classified into regions and will be deformed to form the expressions:



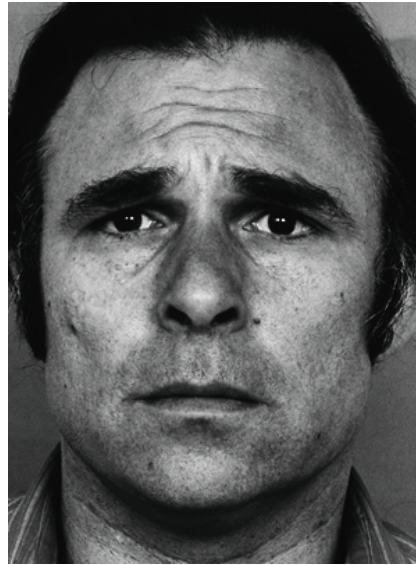
Right after that, I grouped the markers (AAM markers and the points in the lattice) into the regions they mention but filling a bit more the gaps (and of course, not building groups of data that won't be used to create the facial drawing because there is no line to be deformed). These groups were made by considering the shape of the regions in the paper, and running for every shape an algorithm that tested if the point was in/out that certain region.



Once we have the markers, we can start building an expression. I decided to do the sad expression as, according to my experience, it's the face that it's fastest recognizable by people.

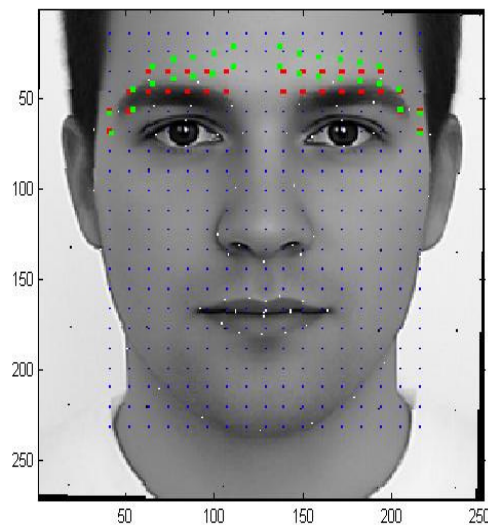
The AUs of this expression have been listed previously in the FACS subchapter.

Let's start with AU1, the *Inner brow raiser* and AU2 *Brow Lowerer* which affect the same regions. This combination has been defined by Ekman and should produce these results (only the remarkable points are shown):



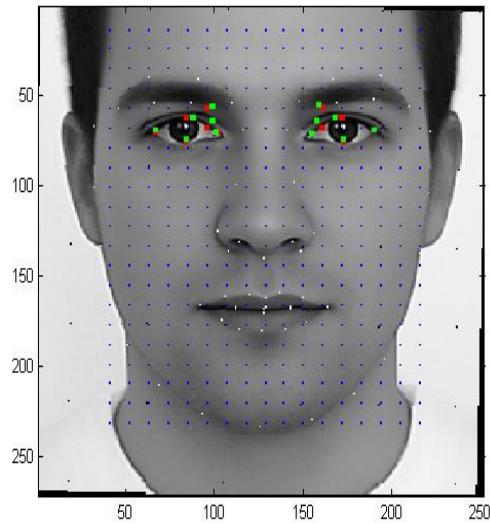
Paul Ekman

The resulting deformation of the points is the following:



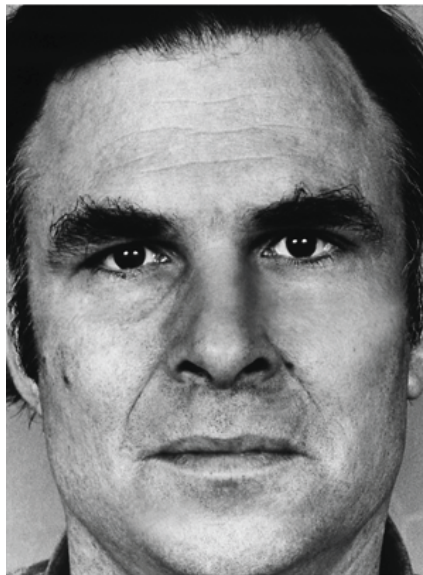
The red points are the original markers and the green points are the deformed ones.

These AUs don't affect only the eyebrows but also the eyes. It makes the eyes look more like a deviated triangle towards the inner brows.



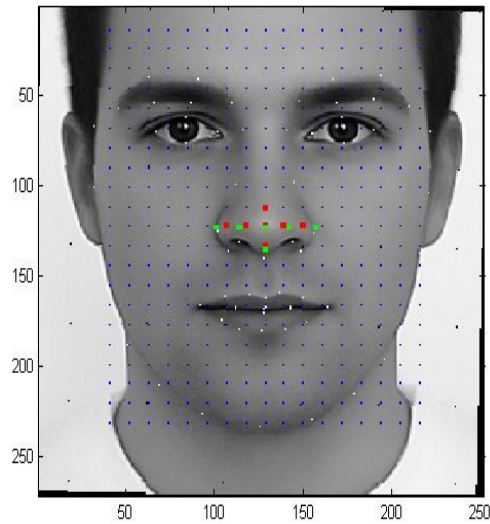
In the eyes in can also be noticed the AU6 *Cheek raiser*, which pushes the lower eyes slightly upwards.

The AU11 *Nasolabial deepender* widens the nose:

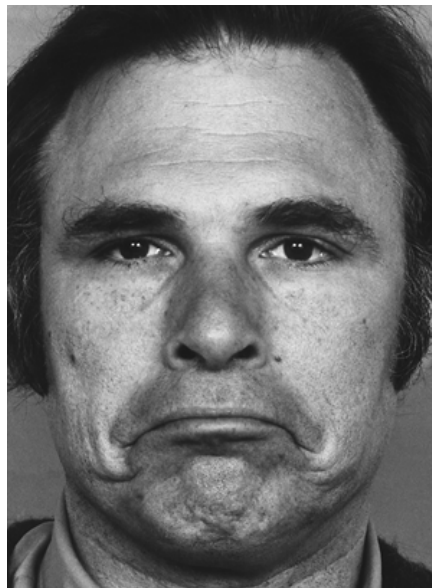


The resulting deformation points are the following:





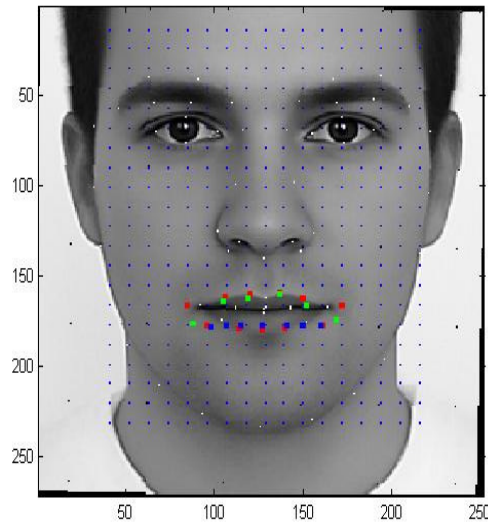
The last two AUs to introduce are AU15 and AU17, *Lip Corner Depressor* and *Chin Raise* respectively. According to FACS, this is what should be seen



*Paul Ekman*

And the deformed set of points is the following:





All the deformed regions that will affect the drawing have been done, so what goes next is finding the coefficients of the rubber-sheet transformations for every region and store them to apply them to another input image.

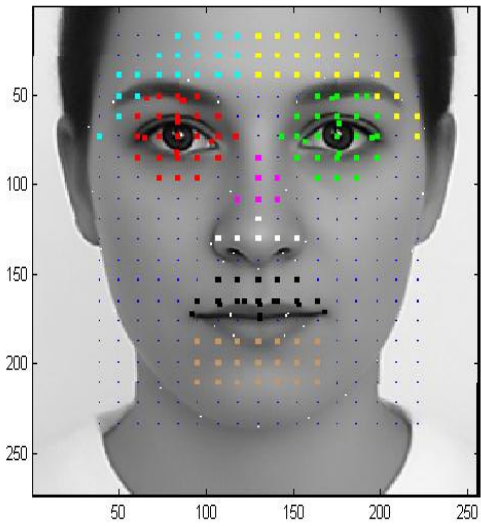
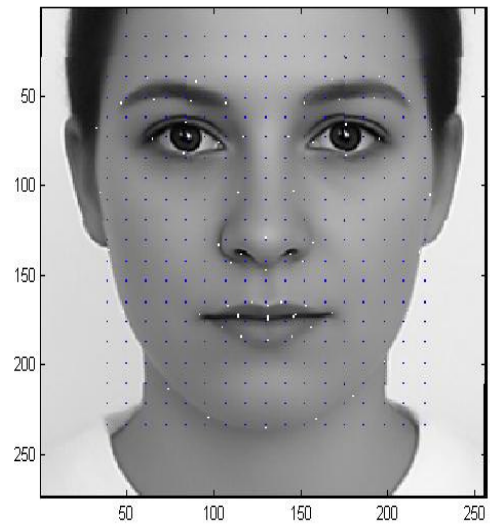
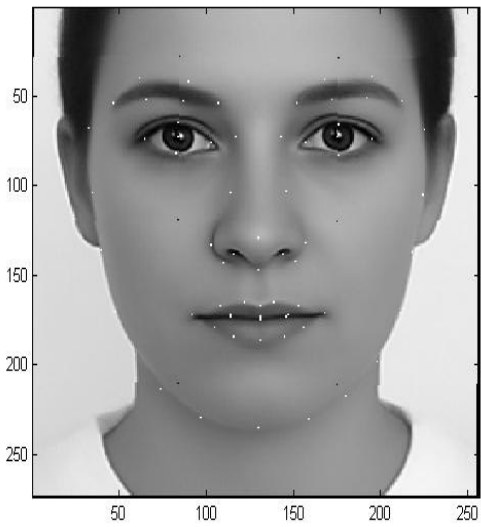
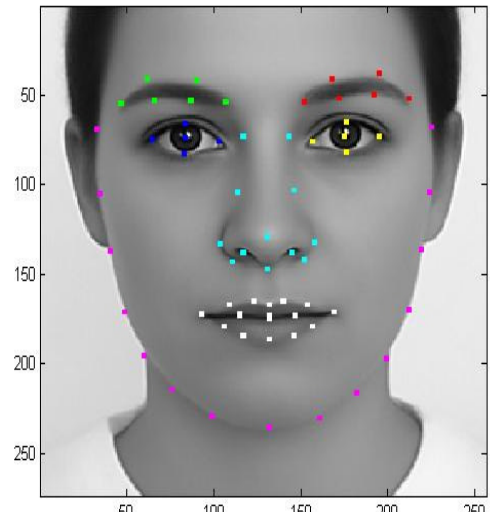
The procedure has already been explained with detail before, so it is just a matter of translating it into Matlab changing the input and output points.

Let's start with the transfer process.

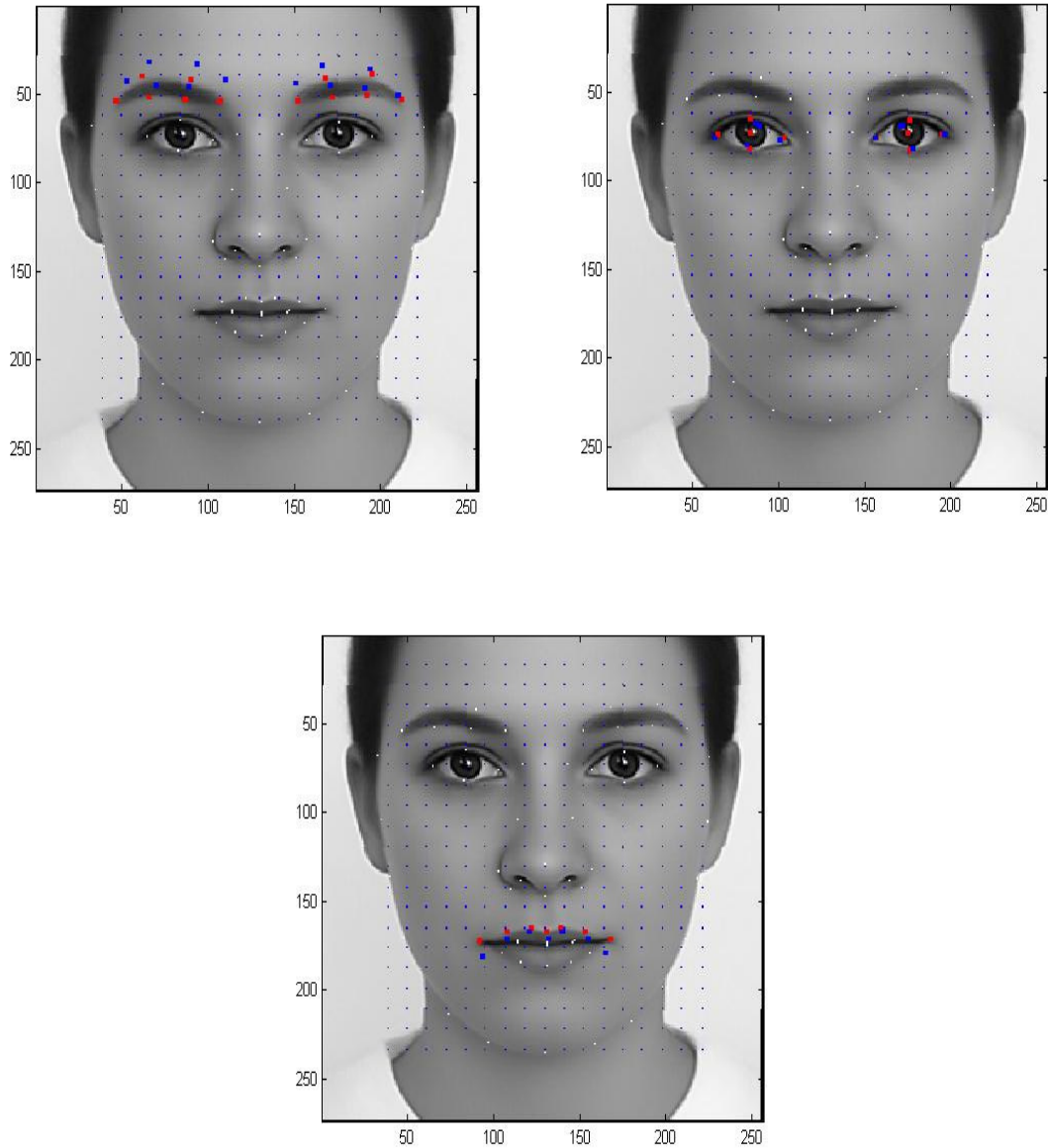
I found another picture in Internet these deformations will be applied to.



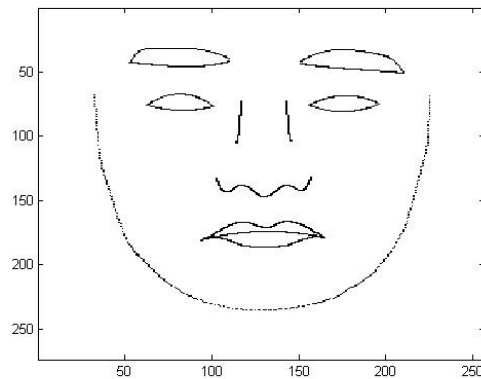
In order to run the deformation equations on this image, we have to detect its facial features, normalize it and put the markers.



With the image this way and the group of markers defining the regions already obtained, we can now apply the rubber-sheet transformations.



The only phase left is the drawing of the lines to form the drawing.  
In the paper, it's not indicated how they draw them, so I decided to use cubic splines.  
Here are the final resulting sad facial components.



- Advantages of the method:

This method suggests a very good idea to express the set of deformations that one may want to transfer. In the paper “*Semi-supervised Learning of Caricature Pattern from Manifold Regularization*” the authors were encouraged to build a manifold due to the results obtained in the literature about the fact that the set of caricatures can be represented in a manifold, and its dimension can be expressed with little error with 17 eigenvectors doing a PCA analysis according to the paper “*Mapping Learning in Eigenspace for Harmonious Caricature Generation*”. All these ideas may also lead to thinking that maybe these deformations can also be described as a polynomial of a certain limited degree. That’s what they try to do in this set of two papers, getting good transfer of the information to new targets. That’s the reason why I felt encouraged to have a try and implement everything following a slightly different set of data that, to my view, was a bit more reliable than what they used, and compare the results.

- Drawbacks and complaints:

This subchapter is not only centered in the last paper, but it also refers to the basic one from 2009, as the authors base almost all the algorithm on it.

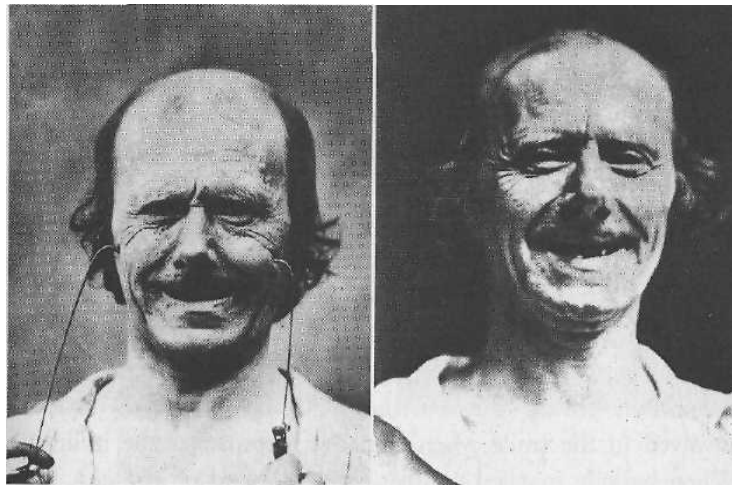
The definition of the regions, according to the authors, has been done looking at how Ekman organized the muscles in the face of a person. However, they don’t mention the fact that these regions correlate with each other, maybe because this way there’s no problem of interaction between the rubber-sheet transformations being applied in turns or at the same time to a single pixel.

I found another problem in this paper here. It’s not said in the paper that the authors were asked to perform the facial expressions following FACS rules, which are very concise. For instance, in the book “*Emotions revealed: recognizing faces and feelings to improve communication and emotional life*” by Ekman, there are instructions to achieve every expression and feel it. Here is an example:

- Drop your mouth open.
- Pull the corners of your lips down.
- While you hold those lip corners down, try now to raise your cheeks, as if you are squinting.  
This pulls against the lip corners.
- Maintain this tension between the raised cheeks and the lip corners pulling down.
- Let your eyes look downward and your upper eyelids droop.

In addition to this, incorrect movements of the face may force some facial regions of the face to move as some muscles provoke the movements of the neighbor, while they should have remained still for a concrete facial expression.

Apart from that, some emotions can't be felt artificially and the result can't be reliable. A clear example of the difference between a face of a person who is trying to perform an expression and a person who is feeling it indeed is the Duchenne smile. Duchenne was a neurologist that was analyzing studying people's facial appearance by electrically stimulating every muscle in the face, and taking pictures of them afterwards. He activated the muscle of the face that makes the lips elongate towards the ears *zygomatic major muscle* and took a picture of it. After that, he told a joke to that man and took another picture. As expected, the pictures were different. There is a muscle used to perform a smile that can be controlled voluntarily: *orbicularis oculi*. This muscle makes the lower part of the eyes go slightly upwards. Here are the two different pictures:



Due to this, I believe that using actors to perform it can't be totally trustworthy.

However, I found still another problem in the method: they do an average on the coefficients found. Bearing in mind the non-linear nature of the rubber-sheet transformations, doing an average may lead to unpleasant and unreliable results.

For this reason, and the lack of application of FACS (even though they mention it) to the expressions when being performed, I wouldn't use this method to transfer facial expressions to avatars/images, as the final images are bound not to be a hundred per cent reliable at all.

Nevertheless, the problems with the expressions are still not over. Some AUs that are essential to form an expression are never captured, because the final image doesn't have any lines there. A clear example would be the *Chin Raiser* (AU=17).



Therefore, some of the final expressions drawn with this limited amount of lines end up being a bit neutral, and if expressive, they indicate a direction at most. Neither they considered these sort of AUs, nor they applied the effects they produce on the other part of the face. For example, almost no difference can be seen between the neutral face and the sad face in the bottom lip, and the bottom lip is normally severely affected by the chin raiser.

The facial features are detected using AAM, and the authors draw the lines of the face and the lines of the background before assembling the two parts. After these transformations, there is no guarantee that the lines coincide, but they surely have to force the continuity.

As for the facial deformation transfer, the sum of all the regions is not the whole face, then, there are some points in the middle that need to be transformed and the authors don't indicate how they do it. Either if they use the closest one, or they do an average of the k-neighbors,...

## Conclusions

The main goal of this thesis was to find some algorithms that had been important in the field and compare them. Unfortunately, there is no point in doing it as the process of creating a caricature is never successfully done by any paper, so I can only compare how much each algorithm provides to the generation of those. In order not to be very repetitive, because the discussions of the methods were done in the previous chapters, I'll mention some of the comments done and will suggest possible improvements to the approaches.

In the first paper, we saw an algorithm that was able to warp an image smoothly according to a facial deformation done. The main contribution of this paper was offering this interpolation process to warp the final image towards the exaggerated facial effect. I don't know if the authors decided to use RBFs as an interpolation algorithm because they thought it was suitable for this purposes, or if it was selected because RBFs have been a general trend for the last five years.

What is essential when choosing an interpolation method is to pay attention to the nature of the data that is going to be approximated, and the authors don't guarantee in any way, that this method was appropriate regarding the input data they had.

Personally, I think that this choice could have been a good choice if the number of points had been more flexible. After getting the final results, I found myself many times trying to move the set of points given to see if, bearing in mind the radial nature of RBFs, a better warped image could be obtained somehow. RBFs seem to be a good interpolation process for 2D field deformations which are not completely linear, but if you limit the amount of center points, you can easily see the circles in the image, what leads to unpleasant results.

Moreover, as I mentioned before, the images used were illuminated and the shadows were warped too. Due to that, incorrect shadowing can be seen in the pictures. A nice way to improve that would be eliminating the illumination of the input image first, and then run the algorithm, as it's better to have a nice smooth picture that has no errors even though it's not completely natural, than returning an image with errors like this one.

In the second paper, we saw an approach that let the caricaturist work freely while achieving a funny digitalized caricature. However, the amount of effort required to put points in their right places didn't compensate for the resulting image. A total of 86 points had to be defined only for the landmarking process. After that you also had to mark where the other control points for every Bézier curve had to go, what made an approximate total of 150 points more. But it's not finished yet; another group of control points to paint the final caricature had also to be set, what added approximately another set of 150 points to put. Regarding that we are talking about a number of almost 400 points to be placed, a method to interactively place them should have been suggested to make the work a bit little for the caricaturist. If this had been done, maybe the balance between amount of work and results could have been a bit more compensated.

And in the third method, we saw a paper that could do nice transfers of deformations from a set of data to input images. I really enjoyed the idea of using rubber-sheet

transformations for this purpose, as in previous paper read, the general idea we to go along this path to find a good way to express the deformation patterns.

This methodology cannot only be applied to facial expressions transfer to caricatures, but it could also be applied to other kind of models with the same behavior as the deformation trends. Transfer of skin strokes to faces come to my mind first, as adding details to surfaces is a very tedious work, and natural faces are not symmetrical. This can be done due to the non-limited nature of the rubber-sheet transformations; in this paper we used the quadratic model, but other cubic models following the same criteria can be constructed for 3D.

As for the rest of the paper, I must say that I considered that the authors tried to find an excuse with which to use the model of interpolation that they had built. The authors were not very careful when gathering the facial expressions, when they defined the facial regions, nor when they approximated the deformations (they did an average of the coefficients of all the actors they hired). It could easily be seen in the results that the faces were not expressive (maybe because of the average done to the coefficients, maybe due to fact that they didn't tell the actors to perform the expressions following FACS rules). Some of the faces didn't even show a trend to the expression they were trying to achieve.

I had a try taking into account that I habitually work with facial expressions, and gave a more accurate input for the method. The results proved that the rubber-sheet transformations were suitable for these purposes.

My general opinion of the amount of papers read is that most of them used interpolation method either to spread the transformation over an image, to transfer the deformation to other inputs, or to build the caricature itself. However, almost none of them added a discussion on how appropriate was the method chosen to approximate the data, what led to strange results sometimes (it's the case of the first implemented paper), and to limitations some other times (it's the case of the second method implemented). In the case of the third paper, the discussion on it had been vaguely done by other authors in other papers, but the authors of this third paper didn't mention it. So I must say that I consider it to be a lack of control over the wide variety of interpolation/approximation methods that exist nowadays.

I personally prefer the data base methods as they offer faster results to the end user apart from the fact that they require, if they do, little professional skills in order to get a good caricature; but if the results are going to be of arbitrary nature due to having been careless in the process, the work done is not worth it.



## Bibliography

1. Brennan, S.: *Caricature generator*. Massachusetts Institute of Technology, Cambridge (1982)
2. Akleman, E., 1997. *Making caricatures with morphing*. In Proceedings of the Art and Interdisciplinary Programs of SIGGRAPH'97. ACM Press: New York, 1997; 145.
3. M. Tominaga, S. Fukuoka, K. Murakami and H. Koshimizu, *Facial caricaturing with motion caricaturing in PICASSO system*, IEEE/ASME International Conference on Advanced Intelligent Mechatronics, 1997, pp. 30.
4. B. Gooch, E. Reinhard, and A. Gooch, *Human facial illustrations: Creation and psychophysical evaluation*, ACM Trans. Graph., vol. 23, no. 1, pp. 27–44, 2004.
5. Ping Wei, Yuehu Liu, Yuanqi Su, *A Method for Deforming-Driven Exaggerated Facial Animation Generation*, Cyberworlds, International Conference on, pp. 648-652, 2008 International Conference on Cyberworlds, 2008
6. W. C. Luo, P. C. Liu, and M. Ouhyoung, *Exaggeration of Facial Features in Caricaturing*, In International Computer Symposium, Hualien, R.O.C, December 2002.
7. H. Chen, Y.Q. Yu, H.Y. Shum, S.C. Zhu, and N.N. Zheng, *Example Based Facial Sketch Generation with Non-Parametric Sampling*, Proc. IEEE Int'l Conf. Computer Vision, vol. 2, pp. 433-438, 2001.
8. Lin Liang , Hong Chen , Ying-Qing Xu , Heung-Yeung Shum, *Example-Based Caricature Generation with Exaggeration*, Proceedings of the 10th Pacific Conference on Computer Graphics and Applications, p.386, October 09-11, 2002
9. Pei-Ying Chiang, Wen-Hung Liao, Tsai-Yen Li: *Automatic Caricature Generation by Analyzing Facial Features*, 2004 Asian Conference on Computer Vision, Jeju Island, Korea, Jan 27-30,2004.
10. Yu-Lun Chen, Wen-Hung Liao, and Pei-Ying Chiang. *Generation of 3d caricature by fusing caricature images*. In Proceedings of IEEE International Conference on Systems, Man and Cybernetics, pages 866 – 871, 2006.
11. Timothy F. Cootes , Gareth J. Edwards , Christopher J. Taylor, *Active Appearance Models*, IEEE Transactions on Pattern Analysis and Machine Intelligence, v.23 n.6, p.681-685, June 2001 [doi>10.1109/34.927467]

12. Liu, J., Chen, Y., Gao, W.: *Mapping Learning in Eigenspace for Harmonious Caricature Generation*. In: 14th ACM International Conference on Multimedia, Santa Barbara, USA, October 22-27, 2006, pp. 683–686 (2006)
13. S. Wold, K. Esbensen and P. Geladi, *Principal component analysis*, Chemometrics and Intelligent Lab. Syst **2** (1987)
14. Chen, W., Yu, H. and Zhang, J. J., 2009. *Example Based Caricature Synthesis*. In: CASA 2009: Proceedings of Conference on Computer Animation and Social Agents. Den Haag, Netherlands: CIP Gegevens Koninklijke Bibliotheek
15. Richmond, T.: *How to draw caricature*. <http://www.tomrichmond.com>, 2008
16. Dubuisson, M. P. and Jain A. K.: *A Modified Hausdorff Distance for Object Matching*. In Proc. of Int'l Conf. on Pattern Recognition (1994), Jerusalem, Israel, pp566-568.
17. Lei Wei, Rui Mo, Wei Gao, Yi Zhu, Zhenyun Peng, Yaohui Zhang: *Application of Facial Feature Localization using Constrained Local Models in Template-based Caricature Synthesis*. Image and Signal Processing, 2009. CISP '09. 2nd International Congress in Tianjin
18. I. Matthews and S. Baker, *Active appearance models revisited*, International Journal of Computer Vision, vol. 60, pp. 135-164, 2004.
19. A. Blake and M. Isard. *Active Contours: The Application of Techniques from Graphics, Vision, Control Theory and Statistics to Visual Tracking of Shapes in Motion*. Springer-Verlag, London, 1998
20. Pan Jianjiang, Zheng Jianmin, Yang Xunnian, *Locally Constrained Deformation for Digital Images[J]*, Journal of Computer-aided Design & Computer Graphics, Vo l. 14, No. 5, May, 2002: 385-388
21. J Liu, Y Chen, J Xie, X Gao, W Gao, *Semi-supervised Learning of Caricature Pattern from Manifold Regularization*, Advances in Multimedia Modeling. B. Huet et al. (Eds.): MMM 2009, LNCS 5371, pp. 413–424, Springer-Verlag Berlin Heidelberg, 2009
22. Roweis, S.T., Saul, L.K.: *Nonlinear dimensionality reduction by locally linear embedding*. Science 290, 2323–2326 (2000)
23. Cootes, T.F., Taylor, C.J., Cooper, D., Graham, J.: *Active shape models—their training and application*. Computer vision and image understanding 61(1), 38–59 (1995)

24. Belkin, M., Niyogi, P., Sindhvani, V.: *Manifold Regularization: a Geometric Framework for Learning from Labeled and Unlabeled Examples*. Journal of Machine Learning Research 7, 2399–2434 (2006)
25. Yu, H. and Zhang, J. J., *Caricature Synthesis Based on Mean Value Coordinates*. In: CASA 2010: Proceedings of the 23rd International Conference on Computer Animation and Social Agents , 31 May-2 June 2010, Saint-Malo, France.
26. M Obaid, R Mukundan, M Billinghamurst, *Rendering and Animating Expressive Caricatures*, Third IEEE International Conference on Computer Science and Information Technology, IEEE ICCSIT2010. 9-11 July 2010, Chengdu, China.
27. J. Canny, *A computational approach to edge detection*, IEEE Trans. Pattern Anal. Mach. Intell., vol. 8, no. 6, pp. 679–698, 1986.
28. Weisstein, Eric W. *Least Squares Fitting--Polynomial*. From MathWorld--A Wolfram Web Resource.  
<http://mathworld.wolfram.com/LeastSquaresFittingPolynomial.html>.
29. P. Ekman and W. Friesen. *Facial Action Coding System: A Technique for the Measurement of Facial Movement*. Consulting Psychologists Press, Palo Alto, 1978
30. Obaid, M., Mukundan, R., Billinghamurst, M., and Sagar, M., 2009. *Facial expression representation using a quadratic deformation model*. In proceedings of the International Conference on Computer Graphics, Imaging and Visualization (August 11 - 14, 2009). CGIV. IEEE Computer Society.